California State University, San Bernardino
CSUSB ScholarWorks

Theses Digitization Project | John M. Pfau Library

2005

Hazweb: An Internet approach to mapping hazardous locations

Kevin Gonzago

Follow this and additional works at: https://scholarworks.lib.csusb.edu/etd-project

 Part of the Geographic Information Sciences Commons

Recommended Citation

Gonzago, Kevin, "Hazweb: An Internet approach to mapping hazardous locations" (2005). *Theses Digitization Project*. 2747.
https://scholarworks.lib.csusb.edu/etd-project/2747

This Project is brought to you for free and open access by the John M. Pfau Library at CSUSB ScholarWorks. It has been accepted for inclusion in Theses Digitization Project by an authorized administrator of CSUSB ScholarWorks. For more information, please contact scholarworks@csusb.edu.

HAZWEB: AN INTERNET APPROACH TO MAPPING HAZARDOUS LOCATIONS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Kevin Gonzago

June 2005

HAZWEB: AN INTERNET APPROACH TO MAPPING HAZARDOUS LOCATIONS

A Project

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Kevin Gonzago

June 2005

Approved by:

[redacted]

Dr. Richard Botting, Chair, Computer Science

June 7, 2005

Date

[redacted]

Dr. Ernesto Gomez

[redacted]

Dr. Keith Schubert

© 2005 Kevin Gonzago

ABSTRACT

What is a Geographical Information System (GIS)? A GIS can be thought of as

> ...an arrangement of computer hardware, software, and geographic data that people interact with to integrate, analyze, and visualize the data; identify relationships, patterns, and trends; and find solutions to problems. (GIS Dictionary)

A GIS can be used in many different sectors. These can range anywhere from business, emergency, government, environmental, to defense. This paper focuses on how GIS can be used with the Internet to locate a provided address or a latitude/longitude position and subsequently determine any hazards that fall within a specified distance to this position.

ACKNOWLEDGMENTS

I thank the faculty of Computer Science department for giving me an opportunity to pursue my M.S. in Computer Science at California State University, San Bernardino. I express my sincere appreciation to my graduate advisor, Dr. Richard Botting who offered me this project and directed me through this entire effort. I also thank my other committee members, Dr. Ernesto Gomez and Dr. Keith Schubert for their valuable input. Lastly, I want thank my wife, Heather, for all her support and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Introduction

GIS technology has reinvented how people think about spatial data analysis. What first started out as just a small niche used primarily by the federal government (for example, the US Census Bureau and the US Geological Survey), ended up becoming a widespread industry used throughout the world in an array of various areas. The tax assessor maps provided by a county government were undoubtedly created using GIS technology. This technology can also be used to determine the best location to build a new little league baseball field. A call made to a 911 dispatcher taps into a GIS to route the nearest emergency personnel to a given address. Address data alone cannot provide the location where emergency personnel need to go (or a little league field for that matter.) It is the actual spatial data, the latitude and longitude coordinates, of these addresses that provide the foundation of a GIS.

GIS technology has evolved significantly in the past 30 years. It is no longer a technology geared

specifically toward computer scientists and/or geographers, but rather to everyday computer users. This may be due to the arrival of personal computers and the Internet.

This paper will not focus primarily on the theory of GIS, but rather look at how a GIS can be used to perform a specific task via the Web. This task determines proximity to hazardous locations given a specified input. The input can be given as an address or as a latitude/longitude position. Using the resulting point location, a choice is provided to the end user that determines hazards within either a one, five, or 10-mile radius. The hazards used in this project are flood zones, fire threat, and earthquake faults. The geographic area covers the extent of cities San Bernardino, Redlands, and Yucaipa, California.

1.2 Purpose of the Project

The purpose of the project was to develop a web application that would map addresses or coordinate information and then find any hazardous areas that may fall within a given distance to this location. All hazardous locations found are flagged to the end user.

This type of information can be extremely useful for various user groups. For example, a realtor looking to see a specific house can get tertiary information regarding its proximity to the nearest earthquake fault line. Insurance companies could use applications such as this to determine a homeowner's risk to possible fire, flood, or earthquake. The application is designed to be user-friendly. Keeping this in mind, the web form design was uncomplicated with easy-to-read map data and tools used to navigate throughout the map's extent.

1.3 Context of the Problem

HazWeb is a web application designed for ease of use for end users such as real estate agents and developers, existing or prospective homebuyers, or insurance adjusters. It would be accessible via a standard web browser such as Microsoft's Internet Explorer, Mozilla FireFox, Netscape, or Opera. The end user would only have to type the specified web address into the URL and follow the directions on the main page. The page prompts for either an address or a coordinate location. Once the user enters this, a point location is generated and displayed in a map. They can then type in a distance, in miles, to

determine any hazards that may fall within this distance. If any hazards are detected, their features in the map are selected and their names displayed.

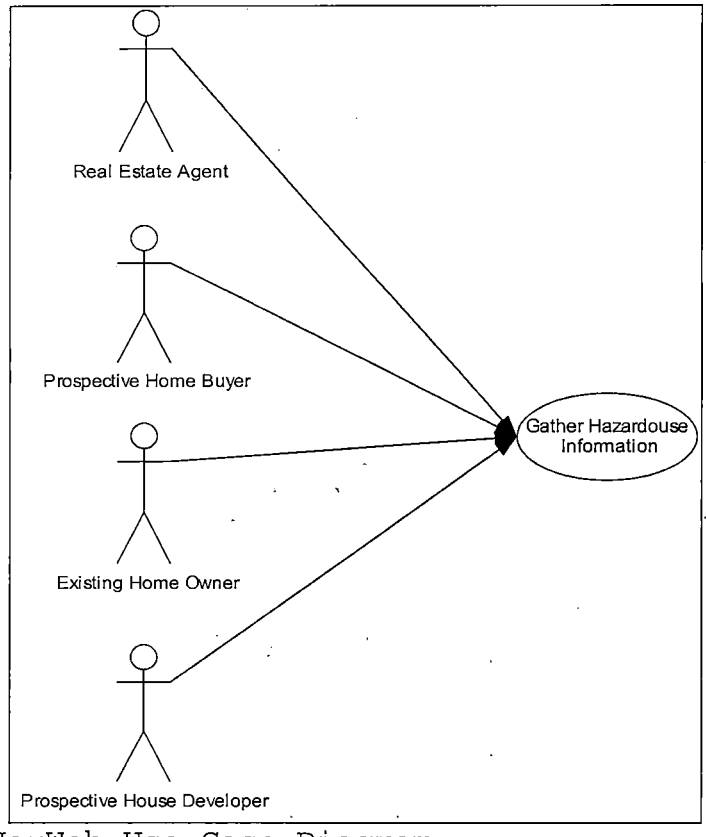


Figure 1. HazWeb Use Case Diagram

The cities of San Bernardino, Redlands, and Yucaipa, California were chosen for this project. This limit was set since the actual amount of data needed to work at a national scale is extremely large and costly. In a real-world scenario that covers a much larger scale, a dedicated server would be needed to store all of the

data. For the purpose of this project, a small test area was used and deemed sufficient.

The data needed for this project includes roads and hazard data such as flood zones, potential fire threat, and earthquake faults. This data fell within the spatial extent of the three cities stated above. Much of this data was freely downloadable from various government web sites.

1.4 Significance of the Project

Southern California has seen its share of natural disasters within the past few years. The wildfires of 2003 will go down in history as some of the most costly and damaging ever to face the City of San Bernardino and surrounding areas. The above-average rain season of 2004-2005 caused massive mudslides and flash floods in certain areas throughout the region. With the increasing onset of people relocating to the Inland Empire region, it is imperative to have a GIS set up to map, display, and analyze these areas of potential disaster. Realtors, existing and potential homeowners, and insurance adjusters can take advantage of GIS technology to pinpoint locations and hazardous proximity. Accessing

this technology via the Internet makes it even more convenient for some people that may not have the resources to obtain it otherwise.

1.5 Assumptions

The following assumptions were made regarding the project:

1. The end user would be literate in standard Internet browsing capabilities. They will understand how to open a web browser and access the web site.
2. There must be no network connectivity problems. If problems arise, the web browser would fail to find the HazWeb website. Additionally, there will be adequate amount of network bandwidth. Users will have a Digital Subsciption Line (DSL) network or better when accessing the website.
3. There must be no hardware resource limitation. In general, spatial data is large, and analyzing it can be extremely resource-intensive. The web application hosting the GIS functionality performs server-side

processing. Although the processing of data is performed on the server, it is still a good recommendation to increase the amount of memory on the client machines. This would help in circumstances when working with large datasets. Displaying and refreshing large datasets can consume quite a bit of resources on the client end.

4. Last, we assume the web server and/or database server is running problem-free.

1.6 Limitations

During the development of the project, a number of limitations were noted. These limitations are presented here.

1. Initially, an additional hazard of high wind was to be included with flood zones, fire threat, and earthquake fault lines. After careful consideration, the high wind data was excluded from the project. It will be redundant to include this data since the study area of San Bernardino County is already considered within the high wind area. It was also deemed

extremely doubtful that data of this type was accessible. The other three data are what is considered static data that can be recorded in point, line, or polygon format. High winds are always changing, thus making the data temporal and difficult to acquire. This makes mapping such types of data impossible in a project scenario.

2. The ESRI ArcGIS Server is a licensed technology. Having the project on a CSUSB local/wide area network requires a full license from ESRI. Therefore, this project was developed and will be demonstrated on a single laptop.

1.7 Definition of Terms

The following is a listing and definition of terms used throughout this paper:

ASP.NET - Active Server Pages used in combination with Microsoft's .NET framework.

ESRI - Environmental System Research Institute; company that produces GIS products such as ArcGIS Desktop and ArcGIS Server.

AO - ArcObjects; ESRI's proprietary object model used in developing a number of GIS products. It is also what is used in application development for map query, display, and analysis.

GUI - Graphical User Interface;

> A software display format of the input and output of a program that lets the user choose commands by pointing to icons, dialog boxes, and lists of menu items on the screen, typically using a mouse. This contrasts with a command line interface where communication is accomplished via the exchange of strings of text. (GIS Dictionary)

GIS - Geographical Information System;

> A GIS can be thought of as an arrangement of computer hardware, software, and geographic data that people interact with to integrate, analyze, and visualize the data; identify relationships, patterns, and trends; and find solutions to problems. (GIS Dictionary)

HazWeb - Hazardous locations website; the project and purpose for this paper

HTML - Hyper Text Markup Language;

A coding language that is a subset of SGML and is used to create Web pages for publication on the Internet. HTML is a system of tags that define the function of text, graphics, sound, and video within a document, and is now an Internet standard maintained by the World Wide Web Consortium. (GIS Dictionary)

HTTP - Hyper Text Transfer Protocol; the client/server protocol that defines how messages are formatted and transmitted on the World Wide Web.

RDBMS - Relational Database Management System;

A type of database in which the data is organized across several tables. Tables are associated with each other through common fields. Data items can be recombined from different files. In contrast to other database structures, an RDBMS requires few assumptions about how data is related or how it will be extracted from the database. (GIS Dictionary)

SOM - ArcGIS Server Object Manager; a service used to manage the SOCs.

SOC - ArcGIS Server Object Container; machines dedicated to host the server objects that are managed by the

SOM. Each SOC machine is capable of hosting multiple SOC processes. "A SOC process is a process in which one or more server objects are running" (Badar, Cameron et al., p. 723).

URL - Uniform Resource Locator;

> A standard format for the addresses of Web sites. The first part of the address indicates what protocol to use, while the second part specifies the IP address or the domain name where the Web site is located. (GIS Dictionary)

1.8 Organization of the Thesis

The thesis portion of the project was divided into six chapters. Chapter One provides software requirements specification, an introduction to the context of the problem, purpose of the project, and significance of the project, limitations, and definitions of terms. Chapter Two consists of the software design. Chapter Three documents the steps used in testing the project. Chapter Four presents the maintenance required from the project. Chapter Five presents the users manual from the project. Chapter Six presents conclusions drawn from the development of the project. The Appendices containing the

project follows Chapter Six. Finally, the references for the project are presented.

CHAPTER TWO

SOFTWARE DESIGN

2.1 Introduction

Chapter Two consists of a discussion of the software design. A broad overview will be given discussing the objective of the project and the design specifications that were incorporated with it. Importance is placed on ease and use of the application to the end user, robustness of GIS analysis performed on the server, and overall quality and reliability of this application to the general public.

2.2 Preliminary Design

This project implements a web application used to access Server objects needed to perform address matching and hazard mapping. Thus, the components needed to implement this project are a web server, ArcGIS Server, GUI components built into the web application, and an interface Application Programming Interface (API) to programmatically access the ArcGIS Server objects.

2.3 Detailed Design

The HazWeb website is has 3-tier architecture. The first tier is the client that displays the GUI in a web browser. The middle tier is the web application server running on a Windows operating system. This web application server uses ASP.NET, which supports C#.NET. It communicates with the client via HTTP. The last tier is the GIS Server, which houses the SOM and SOC. The SOC is responsible for hosting the ArcObjects components needed to perform the location and proximity analysis.

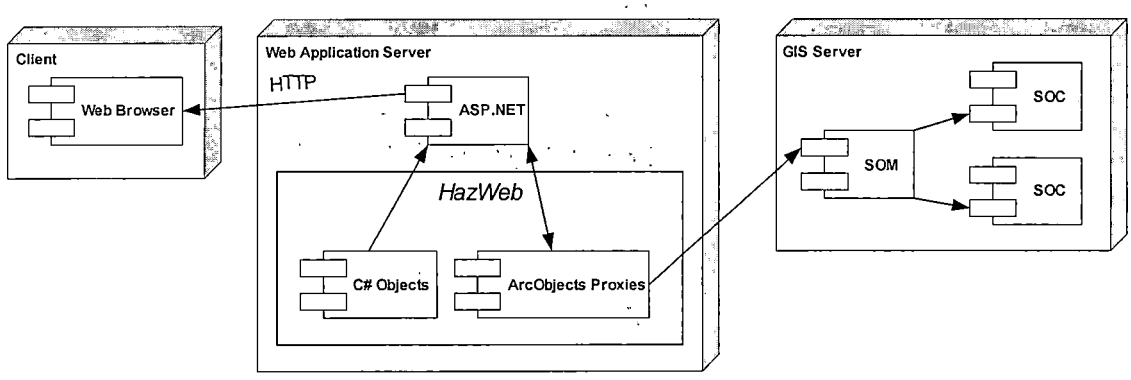


Figure 2. Deployment Diagram of HazWeb Geographic Information System Application

The components used to build HazWeb were chosen due to the following reasons:

1. Web browsers - these are needed to access the Internet. Computers, whether they are used for personal or business use, are usually equipped

with an Internet web browser. Some of the browsers in the project can be downloaded free-of-charge, e.g. Mozilla FireFox, whereas others require licensing. Microsoft Internet Explorer comes provided with Microsoft Windows Operating Systems.

2. Web application - The web application will be implemented using C#.NET. C#.NET is a proprietary language of Microsoft, and it made its debut in 2001.

3. Web application server - the web application server will be implemented using Active Server Pages (ASP) in the .NET environment running off of Internet Information Server (IIS). The server hosts web applications and web services that use the objects running in the ArcGIS server.

4. ArcGIS Server - ArcGIS Server is an object server that manages a set of ArcGIS Server objects. These Server objects are software objects that serve a GIS resource such as a map or a locator. It hosts and runs server objects. The ArcGIS Server consists of a SOM and one or

more server object containers, SOCs. The GIS Server is proprietary software of ESRI.

The following subsections discuss the various facets of this project with detail describing the importance of each.

2.4 Overview of ArcObjects

ESRI developed a suite of GIS products. These products are all built and extended using software components called ArcObjects. ArcObjects are an integrated collection of cross-platform GIS software components that are client and server ready. (ArcGIS Server Administration and Developer Guide)

ArcObjects makes use of the Microsoft Component Object Model (COM). COM is often thought of as simply specifying how objects are implemented and built in memory and how these objects communicate with one another. While this is true, COM also provides a solid infrastructure at the operating system level to support any system built using COM. On Microsoft Windows operating systems, the COM infrastructure is built directly into the operating system. (Badar, Cameron, et al, p. 28)

ArcGIS Server is built on ArcObjects. Since ArcObjects are COM-based objects, interoperability needs to be implemented in order for the .NET web application to work correctly. ESRI has created .NET Primary Interoperability Assemblies (PIA's) that can be seen analogous to their COM type library equivalent. These PIA's are used to obtain access to the underlying COM ArcObjects. Any type of ArcGIS Server development is performed using ArcObjects, and therefore must go through these .NET wrappers. The PIA's get installed with the installation of ArcGIS Server. It may be easier to understand this by explaining some of the basic concepts behind the .NET framework. The next section will go into some detail on this.

2.5 Overview of the .NET Framework

.NET is a new technology designed to change the way software applications are written. This technology is geared towards the efficient development of client and server Internet applications. The new API is also very extensive, providing everything from standard Windows application classes to low-level Web services. .NET also includes a new set of easy-to-use object-oriented

programming languages such as VB.NET and C#. Lastly, this technology is not based on COM; it is a brand new architecture and set of standards for developers (Kelly Hutchins and Jess Borrevik, p. 1-15).

From a developer's point of view, one of the greatest benefits of using .NET is working with the new Visual Studio .NET Integrated Development Environment (IDE) and the new languages it supports. The IDE contains many helpful tools to help the developer write applications efficiently. Some of these tools include an embedded help system and a task-managing system.

The languages now offer many benefits as well. First, VB.NET and C# are both fully object-oriented. Both support structured exception handling and type safety. The .NET Framework objects and Common Language Runtime (CLR) also handle memory management. Both languages call the same .NET Framework classes. These classes contain the functionality to do all basic and advanced programming, including things like Web server development. Lastly, they provide one API through which you can work instead of having to consume a number of APIs such as Win32, ole32.dll, etc. (Kelly Hutchins & Jess Borrevik, p. 1-16).

.NET is backwards compatible with most Microsoft technologies, such as COM. On the client side, the Visual Studio .NET environment gives you the ability to import any COM type library, whether standalone or embedded in an ActiveX control. .NET also provides a set of classes that make it possible for .NET-managed code to interact with unmanaged COM objects. These classes make it possible to access any method of any interface on any given COM component. So in general, .NET provides complete client-side support for COM and ArcObjects development (Kelly Hutchins and Jess Borrevik, p. 1-19). Even though ArcObjects is a COM-based model, development in the .NET is possible via the .NET PIA's provided with the ArcGIS Server installation.

The .NET Framework sits as a layer between an operating system and your application providing a simple framework for developing and running code. .NET has two main components: the base class library and the CLR. The base class library is a set of hundreds of managed code types (classes, interfaces, structures, and enumerations) that provide much of the functionality previously provided by the Windows API. The CLR is the .NET execution engine. When managed code is executed, the CLR

starts your code, manages the threads, and provides necessary background services such as security and COM interoperability (Kelly Hutchins & Jess Borrevik, p. 2-5).

The CLR manages and executes code that is written in .NET languages. The CLR manages such things as security, type safety, COM interoperability, and garbage collection. Code that runs under the control of the CLR is called managed code and has benefits such as platform independence, language interoperability, and performance improvement. Code that does not utilize the CLR is called unmanaged code. Unmanaged code, such as Win API libraries and COM objects, do not use or need the CLR execution environment. COM interoperability provides a bridge that allows us to use COM components from managed code and .NET components from COM (Kelly Hutchins & Jess Borrevik p. 2-8).

As mentioned, ESRI provides the necessary components that bridge the gap between .NET and their COM-compliant ArcObjects. It may be helpful to give a definition of what an assembly is in the .NET world. An assembly can be thought of as a logical grouping of types and resources. Assemblies consist of one or more physical files called

modules that can be portable executable files or resources. Assemblies contain three elements: intermediate language code, metadata, and a manifest describing how the elements in the assembly relate to each other and external elements. The metadata is similar to a type library but far more complete, and it has the advantage of always being included in the file that contains the code.

Compilation in .NET is a two-step process. Building the .NET project uses a language-specific compiler to convert the source code to Microsoft Intermediate Language (MSIL). MSIL is a low-level language similar to Java's byte code that can be quickly translated to machine code. It gives portability between operating systems, interoperability between languages, and execution-management features such as memory management and security (Deitel, Deitel, et al., p. 19).

The .NET Framework is freely downloadable from Microsoft's website. It is fairly large, but once it is downloaded, applications written using .NET can then be executed. If a person is familiar with the Java API, they may see this similar to downloading the Java Runtime Environment and running Java applications on top of this.

2.6 Overview of ArcGIS Server

ArcGIS Server provides developers with the ability to build advanced GIS services and Web applications in a server environment. Developers will be able to use ArcGIS Server to deliver advanced GIS functionality to a wide range of users. ArcGIS Server is built from the same modular, scalable, and cross-platform ArcObjects components that comprise the entire ArcGIS system. ArcGIS Server allows developers and system designers to implement a centrally managed GIS. ArcGIS Server's ability to leverage web services makes it ideal for integration with other critical information technology systems, such as relational databases, web servers, and enterprise applications servers.

ArcGIS Server is composed of a SOM, SOC, ArcObjects, and the Server Developer Kit. The ArcGIS Server is what is responsible for hosting, running, and managing server objects. This server can reside on the same machine as the web server. For the purpose of this project, both the web server (IIS) and ArcGIS Server reside on the same machine.

The SOM manages Server objects distributed across one or more container machines. It runs as a background

Any machine that intends on hosting the web application needs to have the Software Developer Kit installed.

2.7 Spatial Data Used in the Project

A portion of the data used in this project was downloaded from various private and public agencies, whereas the remaining data was accessed from ESRI's sample datasets provided in their software products. There are various formats used for spatial data. The format used for this project was the personal geodatabase and shapefile. Both of these formats are proprietary to ESRI. The personal geodatabase is an Access database (.mdb) that holds tabular and spatial data. The spatial information is stored as binary large objects. Shapefile data is file-based and is comprised of three files: .shp, .dbf, and .shx. The first is the file containing all spatial information, the second holds all related attribute, or tabular, information, and the last is the spatial index.

The hazard data was divided into areas depicting fire threat, flood zones, and earthquake fault lines. There are three separate datasets representing each hazard. As mentioned previously, spatial data is not of

much use without associated tabular information. Each of these hazard datasets has associated attribute information. The information is what depicts the level of fire threat, potential flood possibilities, and the type and name of an earthquake fault. The legend displayed in the MapServer object shows how each of these hazards is displayed to the end user.

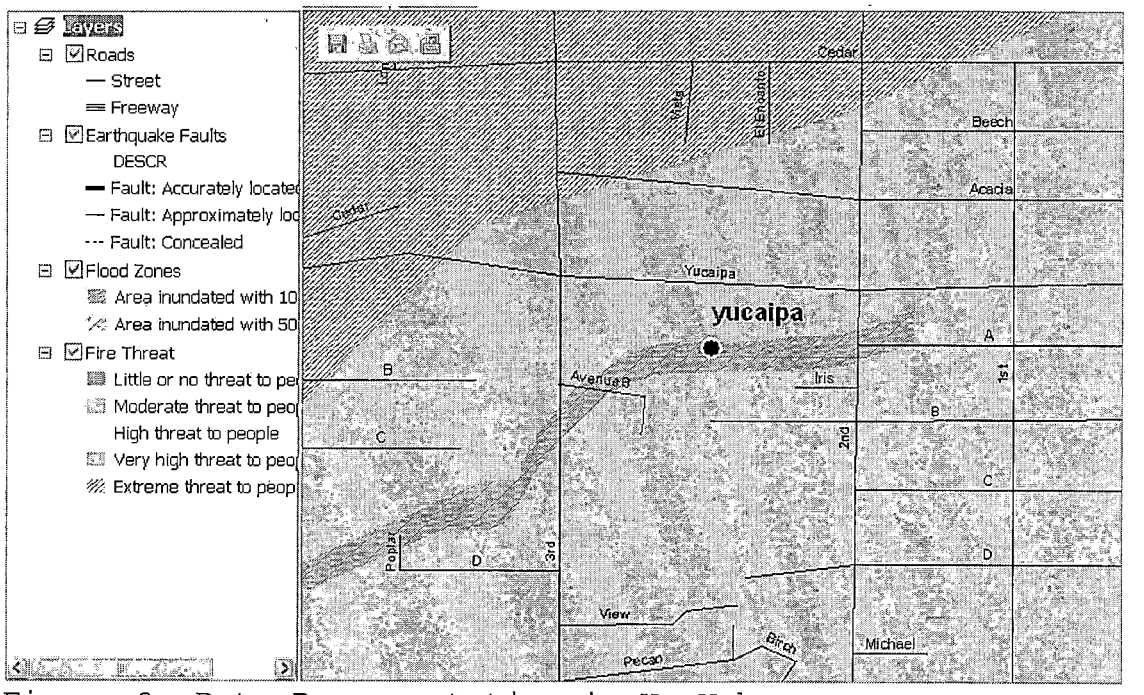


Figure 3. Data Representation in HazWeb

There is also street data that is used to geocode the user's input address or location. Geocoding is another name for address matching. In addition to a MapServer object, various GeocodeServer objects were

created for this project. In order to match a given address to its respective place on a map, it needs to be matched up correctly. Geocoding starts with a textual description of a location and translates that into the X and Y coordinate that can be plotted on a map (GIS Dictionary). The street data, known as reference data, must also have the details of the specific location to find. A GeocodeServer object points to an address locator. The address locator is the main tool for geocoding in ArcGIS Server. It is a file (.loc) that points to resources such as the data referencing the street network.

In order to author the maps and address locators used in the MapServer and GeocodeServer objects, a separate product needs to be installed in addition to ArcGIS Server. ArcGIS Desktop has the applications needed to create a map (.mxd) and a .loc file. The maps are created using a product called ArcMap and the locators are created using another product called ArcCatalog.

Now that a broad overview with definitions has been provided, the following subsection describes the application design for the HazWeb program.

2.8 HazWeb Application Design

Visual Studio .NET was the IDE used in developing the HazWeb application. An ASP.NET project was created using C# as the programming language. Two Web forms are used in this project. The first form is the beginning, or start, page for the user. The GUI in the start form was written using HTML. There are two tabs provided. One for input of address information, the other for input of X and Y coordinates. Figure 4 illustrates the start page.

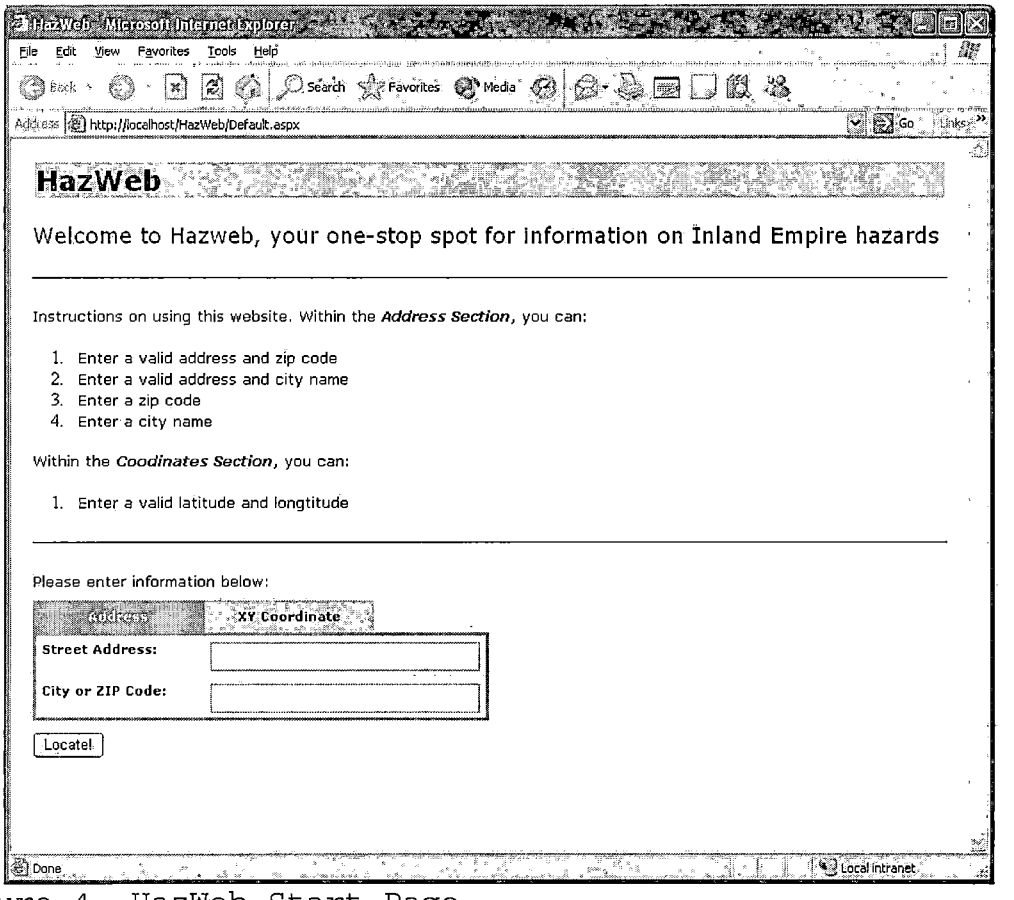


Figure 4. HazWeb Start Page

The default tab is for input of address information. If address information is provided, one of the GeocodeServer objects is used. The type of address specified will determine the precise GeocodeServer object. The other tab, which allows input of coordinate information, creates a point on the fly and displays it in the map.

The user types in either address or coordinate information. If the user chooses an address, they can do it one of four ways:

1. Enter an address with a valid ZIP code.
2. Enter an address with a city name.
3. Enter a city name.
4. Enter a ZIP code.

If the user chooses to enter an XY coordinate location, they would click the "XY Coordinate" tab and enter the XY coordinates. The XY coordinates must be in decimal degree format. For example, -117 for the X coordinate and 34 for the Y coordinate will yield the location of the city of Redlands.

Once the "Locate" command is clicked, a second Web form will load displaying the MapServer object. A point will then appear at the correct location on the map.

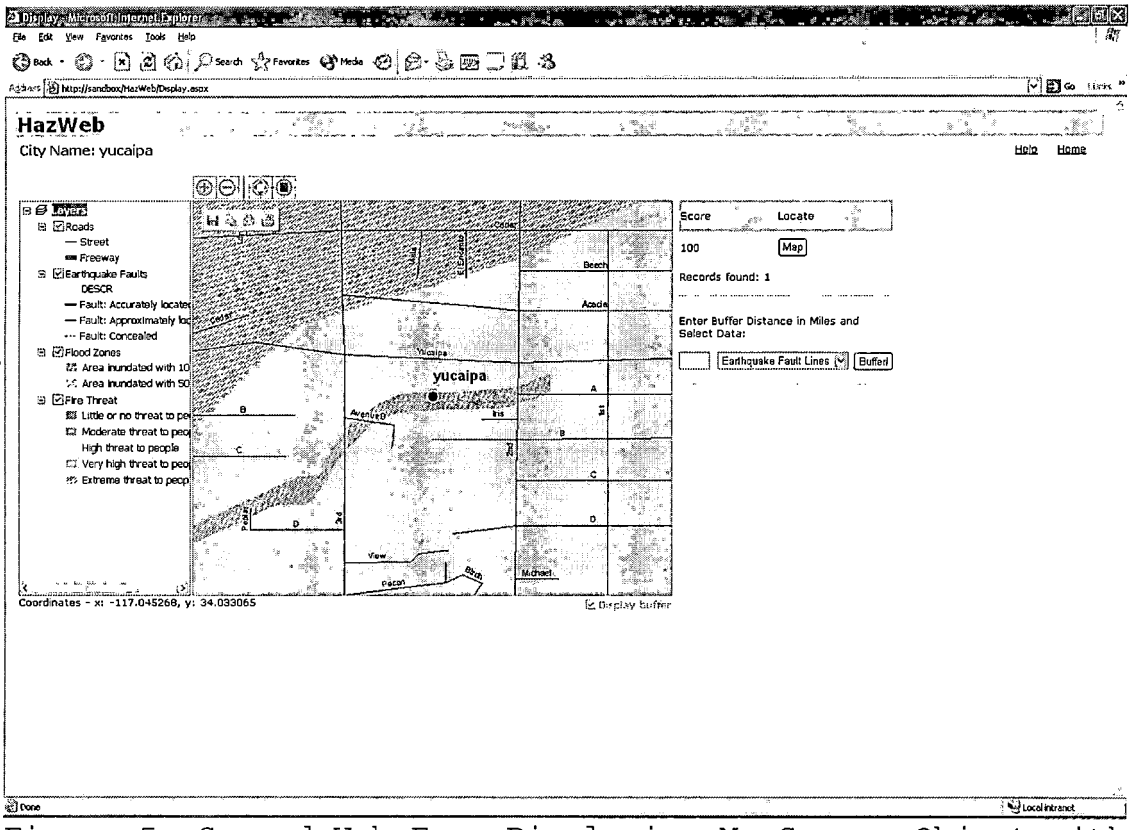


Figure 5. Second Web Form Displaying MapServer Object with Associated Address/Coordinate Point

The second Web form displays a text box that allows input from the user. The user can type in a distance, in miles, to find all hazards that fall within the given distance. A buffer is created around the point location. The user has the option to specify which hazards fall within the buffered region. Figure 6 indicates a two-mile buffer of the flood zone data.

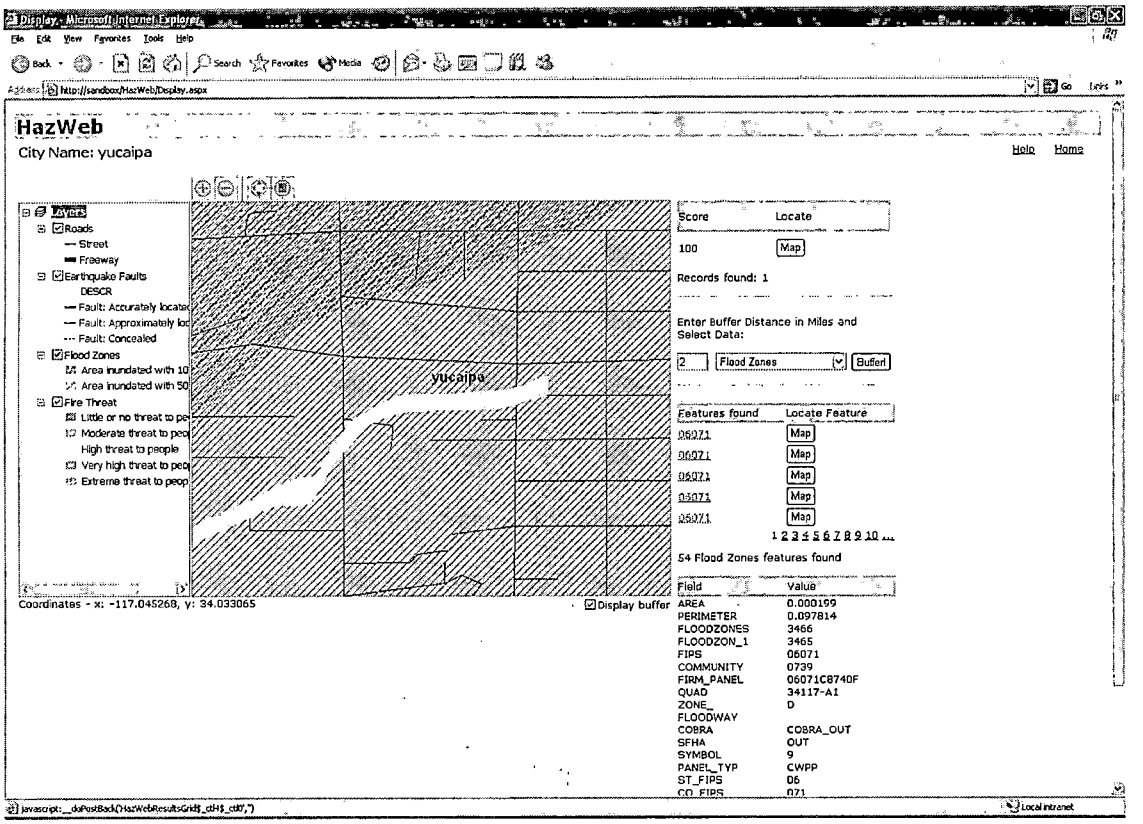


Figure 6. Web Form Displaying Buffered Area Around Specified Address Location

A toolbar with navigation tools is also provided in the web application. This toolbar appears in the second web form that shows the map, address location, and any hazards. It contains tools that zoom in, zoom out, pan, and zoom to the full extent of the map. The user can click on one of these tools, and zoom in/out, or pan based on the OnMouseDown, OnMouseMove, and OnMouseUp click events within the display. The "Zoom to full extent" command will reset the extent of the map to the

original position. ToolTips are provided for these tools. The user can hover over one of them and a description of the tool is displayed. There is also a check box in this form. Once it is cleared, any generated buffers are removed.

2.8 Overview of Interfaces Used

- Internet browser - Microsoft Internet Explorer, Netscape, Mozilla FireFox, Opera
- Operating System - Windows 2000 or XP
- Spatial Database - Microsoft Access
- Web Server - Internet Information Server
- GIS Server - ArcGIS Server
- Integrated Development Environment - Microsoft Visual Studio .NET 2003 (version 1.1)
- Programming language - Microsoft C#.NET

2.9 Summary

The software design of the project was presented in Chapter Two. The HazWeb application was designed to be as straightforward and user-friendly as possible. The only real information that the end user must know is what location they want to find and the distance needed for

hazard data analysis. The MapServer and GeocodeServer objects are relatively uncomplicated to create. The bulk of analysis is performed using the ArcObjects API within the ASP.NET web application. .NET is an object-oriented API, therefore core GIS functionality was written as classes (*.cs files.) In the span of a few keystrokes, an address can be plotted on a map and any nearby hazards can be determined.

CHAPTER THREE

SOFTWARE QUALITY ASSURANCE

3.1 Introduction

Chapter Three documents the software quality assurance process. Specifically, what was done to validate a high level of guarantee that the HazWeb program consistently produced results that met specified guidelines? It is vital in any software application to perform testing. The following sections describe the tests performed at various levels: unit, integration, and system.

3.2 Unit Test Plan

> In computer programming, a unit test is a method of testing the correctness of a particular module of source code. (http://en.wikipedia.org/wiki/integration_testing)

Unit tests for this project were broken down into small, manageable units. The following table illustrates the tests performed and the results achieved for each one.

Table 1. Unit Test Results (Class: AddGraphicElement)

Function	Test Performed	Results
AddGraphicElement	• Adds a graphic element to the custom graphics property.	Passed

Table 2. Unit Test Results (Class: Buffer)

Function	Test Performed	Results
ChangeSymbol	• Changes the symbol of the buffered region.	Passed
ChangeElementSysmbol	• Changes the buffered region symbol to a hashed symbol.	Passed
CreateBuffer	• Creates a buffered region.	Passed

Table 3. Unit Test Results (Class: Default)

Function	Test Performed	Results
Page_Load	• Initializes variables.	Passed
BtnLocate_ServerClick	• Gathers user information and displays the Display.aspx.	Passed
ReleaseSessionVariables	• Clears session variables from the browser session.	Passed
ClearTextBoxes	• Clears values from text box controls.	Passed

Table 4. Unit Test Results (Class: Display)

Function	Test Performed	Results
DynamicColumnAdded	• Property that display columns.	Passed
DynamicResultsColumnAdded	• Property that displays columns added to the results data grid control.	Passed
LoadViewState	• Overloaded function that checks for properties.	Passed
SaveViewState	• Overloaded function that creates the results table.	Passed
GeocodeAddress	• Function that geocode the given address and display it on the map.	Passed
AddColumns	• Function that add columns to the data grid control.	Passed
UpdateUI	• Function that display or hide controls.	Passed
AddResultsColumns	• Function that add columns to the results data grid control.	Passed
ReleaseSessionVariables	• Function that release session variables.	Passed
RetrieveGeocodingRecords	• Function that determine which geocode object to use.	Passed
CandidatesXY	• Function that displays the X and Y coordinates.	Passed
FindFeatureDetails	• Function that displays results of the selected feature.	Passed
DisplayGeocodeRecords	• Function that uses the geocode object to display records.	Passed
GetZoomInfoOnLayers	• Function that retrieves zoom information on each layer.	Passed
ZoomToFeature	• Function that zooms to the selected feature.	Passed
CallErrorPage	• Function that loads the Error page when an error has occurred.	Passed

Function	Test Performed	Results
Page_Load	• Function that initializes all variables.	Passed
HazWebToolBar_CommandClick	• Function that determines which button was clicked on the toolbar.	Passed
HazWebDataGrid_ItemCommand	• Function that determines which cell has been clicked in the data grid control.	Passed
HazWebDataGrid_PageIndexChanged	• Function that displays the next page in the data grid control.	Passed
lnkHome_Click	• Function that clears session variables and displays the Default.aspx.	Passed
HazWebResultsGrid_ItemCommand	• Function that determines which cell has been clicked in the results data grid control.	Passed
HazWebResultsGrid_PageIndexChanged	• Function that displays the next page in the results data grid control.	Passed
btnFind_Click	• Function that creates the buffer and select any features that are in the buffer.	Passed
chkBuffer_CheckedChanged	• Function that displays or hides the buffered region.	Passed

Table 5. Unit Test Results (Class: Error)

Function	Test Performed	Results
Page_Load	• Function that displays the error messages and Error.aspx.	Passed

Table 6. Unit Test Results (Class: Functions)

Function	Test Performed	Results
UnselectAllFeatures	• Function that un-selects all features in the map.	Passed
DisplayMessage	• Function that displays messages.	Passed
ZoomToRoadsExtent	• Function that zooms to the extent of a layer.	Passed
RemoveAllPolygonElementsFromGraphicElements	• Function that removes all polygon elements from the graphic elements property.	Passed
RemoveAllPolygonElementsFromGraphicsContainer	• Function that removes all polygon elements from the graphic container property.	Passed
GetOIDFieldName	• Function that retrieves the name of the OID.	Passed
GetGeometryFieldName	• Function that retrieves the name of the geometry field.	Passed
GetDisplayField	• Function that retrieves that display field name.	Passed
GetLayerName	• Function that retrieves the layer name.	Passed
CreateTableResults	• Function that creates the table results.	Passed
IsMapVisible	• Function that checks if the map is visible.	Passed
SearchFeature	• Function that does a search for the selected feature.	Passed
RemoveAllPointElementsFromGraphicElements	• Function that removes all point elements from the graphic element property.	Passed

Table 7. Unit Test Results (Forms)

Form	Test Performed	Results
Default	• Verify valid values are entered into the text box contols. • Verify the Help link works • Verify the Display.aspx displays correctly after clicking the Locate button.	Passed
Display	• Verify address entered in correct. • Verify zoom in tool works. • Verify zoom out tool works. • Verify pan tool works. • Verify zoom to full extent tool works • Verify Map button zooms to the address • Verify Buffer button creates a buffered region and selects features within it. • Verify user has entered a valid value in the Distance text box control. • Verify Display buffers check box hides and displays the buffered region. • Verify layers in the Table of Contents can be shown or hidden. • Verify results data grid control displays the correct information. • Verify results table control displays the correct information.	Passed

3.3 Integration Test Plan

Wikipedia gives a concise description of integration testing with,

> Unit testing does not catch all errors; this is where integration testing comes into focus. Integration testing is the phase of software testing which software modules are combined and tested as a group. It takes its input modules that have been checked out by unit testing, groups them in larger aggregates, applies tests defined in a test plan, and delivers as its output the integrated system. Inter-process communication is tested. For example, tests are performed across procedure calls and processes. (http://en.wikipedia.org/wiki/integration_testing)

Integration testing was performed for this project. After going through the above-mentioned unit test, they were integrated and tested to verify the stability and precision. The following table shows the specific tests performed at an integration level.

Table 8. Integration Test Results

Integration Test	Tests Performed	Result
Data Entry User enters address and zip code, address and city name, city name, zip code or XY coordinates.	• Verify if user enters a valid zip code. • Verify if user enters valid XY coordinates. • Verify help page is displayed when user clicks on the help link. • Verify information is passed from the Default form to the Display form.	Passed
Display Data	• Verify address entered in correct. • Verify zoom in tool works. • Verify zoom out tool works. • Verify pan tool works. • Verify zoom to full extent tool works • Verify Map button zooms to the address • Verify Buffer button creates a buffered region and selects features within it. • Verify user has entered a valid value in the Distance text box control. • Verify Display buffers check box hides and displays the buffered region. • Verify layers in the Table of Contents can be shown or hidden. • Verify results data grid control displays the correct information. • Verify results table control displays the correct information.	Passed

3.4 System Test Plan

The last stage of the testing process is system testing. "System testing is the phase of software testing where the complete system is tested. It is testing conducted on a complete integrated system to evaluate the competent ness of its specified requirements." (http://en.wikipedia.org/wiki/integration_testing)

System testing of the HazWeb system begins with the following steps:

Table 9. System Test Results

System Testing	Results
1. Install IIS system.	Passed
2. Start up all geocode and map server objects.	Passed
3. Run tests by using real data on all forms.	Passed

3.5 Summary

Testing is an extremely important step in the software development cycle. HazWeb followed the three phases of unit, integration, and system testing to help troubleshoot and fix any areas within the code that were causing errors or just were not working, as they should. The cyclical nature of this testing helps organize the

flow of how the parts of the program worked singularly and as a whole. The testing performed on HazWeb established its accurateness needed to comply with the original system requirements for the project.

CHAPTER FOUR

MAINTENANCE

4.1 Introduction

Chapter Four is a presentation of the installation and maintenance of HazWeb. The subsections outline the steps needed to setup and maintain the HazWeb program. These subsections concentrate on: web server installation, ArcGIS Server software installation and configuration, and ArcGIS Server Object configuration.

4.2 Web Server Installation

IIS is the web server used to host HazWeb. Since this application runs on a Windows XP Operating System, IIS comes standard on Windows Operating Systems. Although it comes standard, the installation process needs to be performed. To do this, follow the steps below.

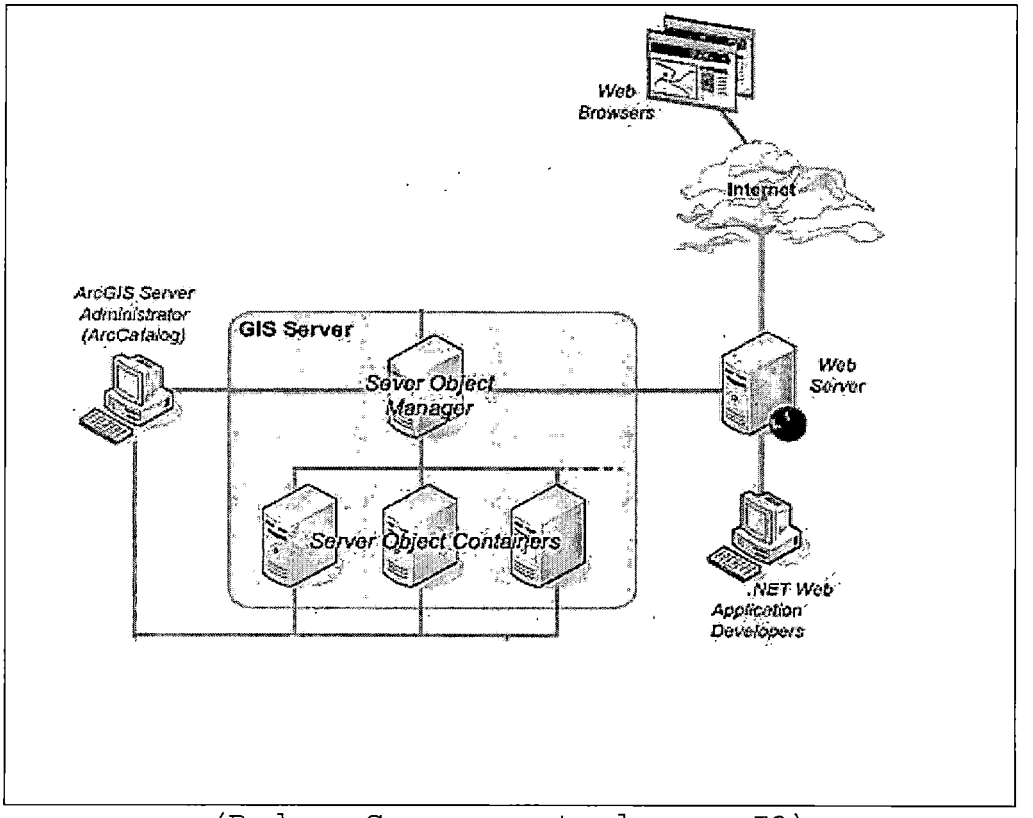
1. Click on Start > Control Panel
2. Add or Remove Programs > Add/Remove Windows Components
3. Click on the check box for IIS
4. Accept all defaults and install.

To test and make certain that IIS was installed correctly, type in http://localhost/localstart.asp to

Internet Explorer's URL address. A welcome page should appear discussing IIS and its functionality.

4.3 ArcGIS Software Installation and Configuration

Before anything is started, ArcGIS Server needs to be installed and configured. The figure below is a representation of how all parts of the HazWeb project fit together.



(Badar, Cameron, et al., p. 73)

Figure 7. HazWeb's Component Organization

There are many possible ArcGIS Server system configurations. It is possible to run all components of ArcGIS Server on a single computer, or to distribute the various components across multiple computers. Configurations will vary depending on the needs of an organization, the available computing resources, and the intended use of the ArcGIS Server system. For the Server Object Containers, processing speed and the ability to handle load are important (Badar, Cameron, et al., p. 50). For the purpose of this project, one computer is used to run all components. A representation of this is provided below.

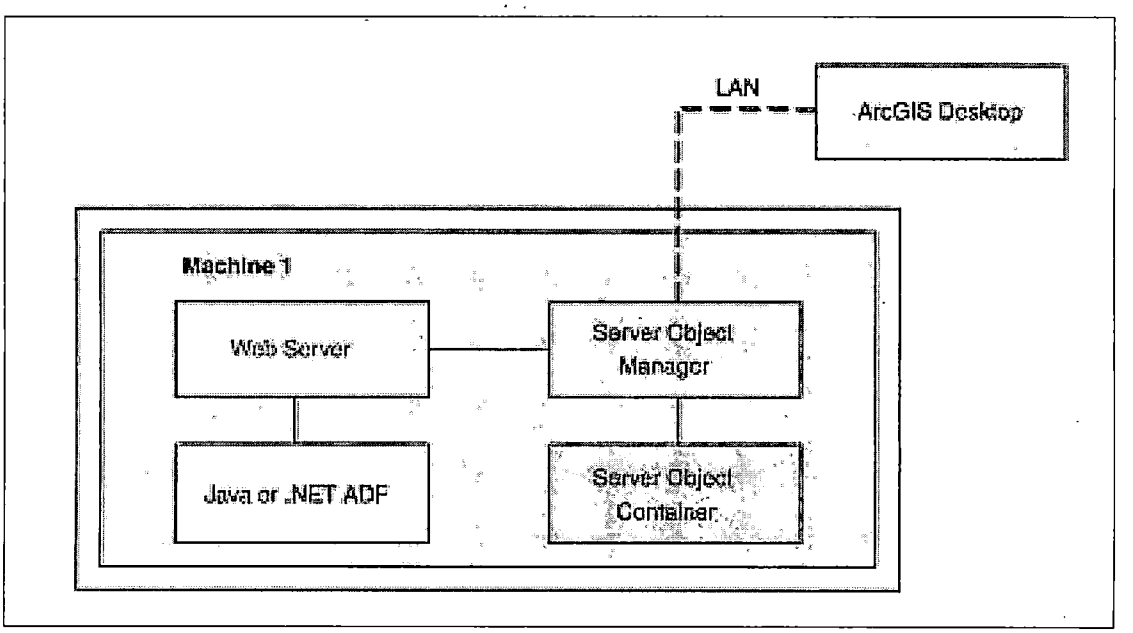


Figure 8. Visual Representation of Single Machine ArcGIS Server Setup

In order to author MapServer and GeocodeServer objects, another product needs to be installed on a machine with access to ArcGIS Server through a Local Area Network. This product, called ArcGIS Desktop, has two applications, ArcMap and ArcCatalog. ArcMap is the authoring product used to create maps hosted in MapServer objects. ArcCatalog is the application used to create address locator (.loc) files provided in GeocodeServer objects. ArcGIS Desktop can be run on any Windows 2000 or XP operating system. After installing ArcGIS Desktop, ArcGIS Server can now be installed and configured.

4.3.1 Install ArcGIS Server

The installation of ArcGIS Server for HazWeb is relatively uncomplicated. It installs on a Windows XP operating system. The installation is provided as a Setup.exe file on the media kit CD. Since the HazWeb GIS Server hosts the SOM and SOC's on one machine, both SOM and SOC installations happen simultaneously. The default location for installation is C:\Program Files\ArcGIS. HazWeb's installation of ArcGIS Server is located here as well.

The system requirements to run ArcGIS Server are at least 512 MB RAM, HazWeb has 1.024 GB installed on its

host machine. In addition to RAM, Microsoft Internet Explorer with at least a version of 6.0 must be installed on the system running ArcGIS Server. HazWeb runs version 6.0 service pack 2.

The following steps illustrate the ArcGIS Server installation process:

1. Insert the ArcGIS Server CD into the CD drive to automatically launch the setup program.
2. During the installation you will be asked to read the license agreement and accept it, or exit if you don't agree with the terms.
3. The ArcGIS Server install allows you to select the features you would like to install.

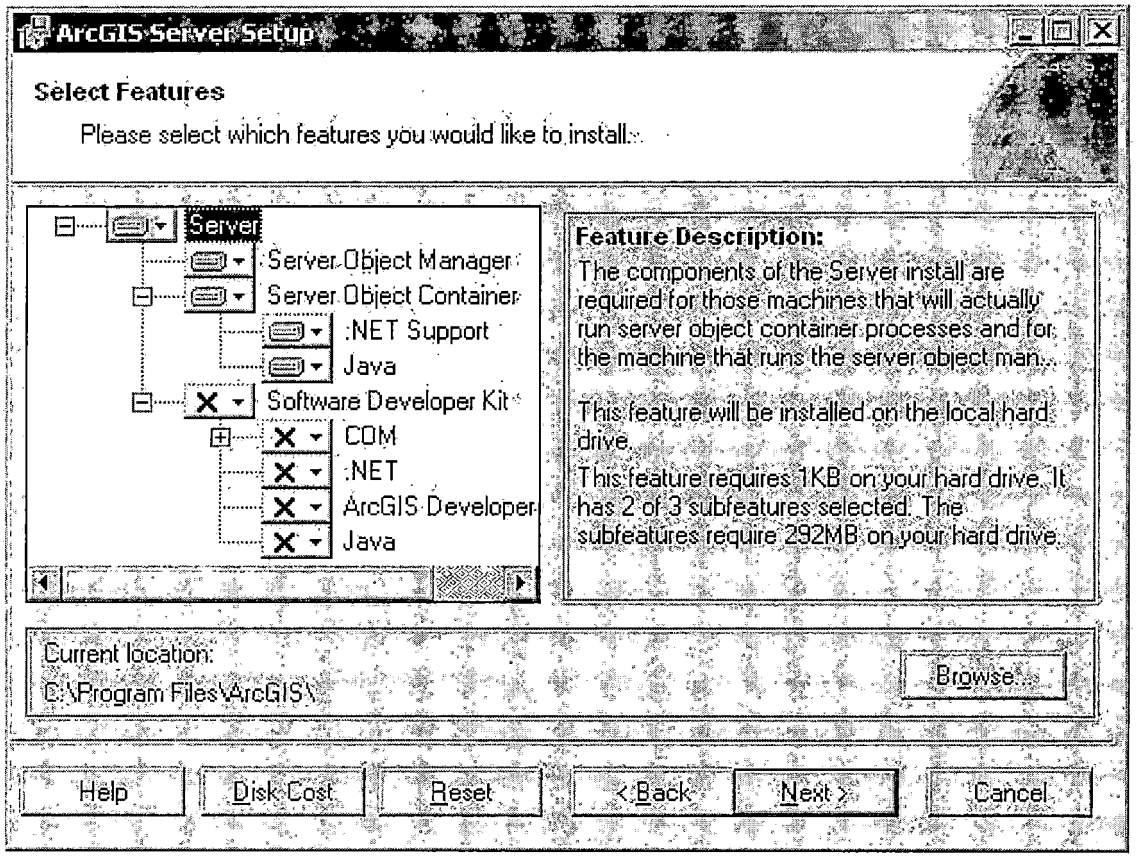


Figure 9. ArcGIS Server Installation Dialog

1. Complete the installation by choosing defaults for remaining options.
2. Run through ArcGIS Server post-installation process.

4.3.2 Post-Server Installation Process

After installation of ArcGIS Server is complete, the user is prompted to run through the Post-Server installation process. This can be done now, or later. If the latter is chosen, it can be accessed by clicking

Start > Programs > ArcGIS Server > Post-Server installation process. A dialog appears asking if the user would like to either: configure ArcGIS Server, authorize ArcGIS Server for use, or do both. Both check boxes should be clicked.

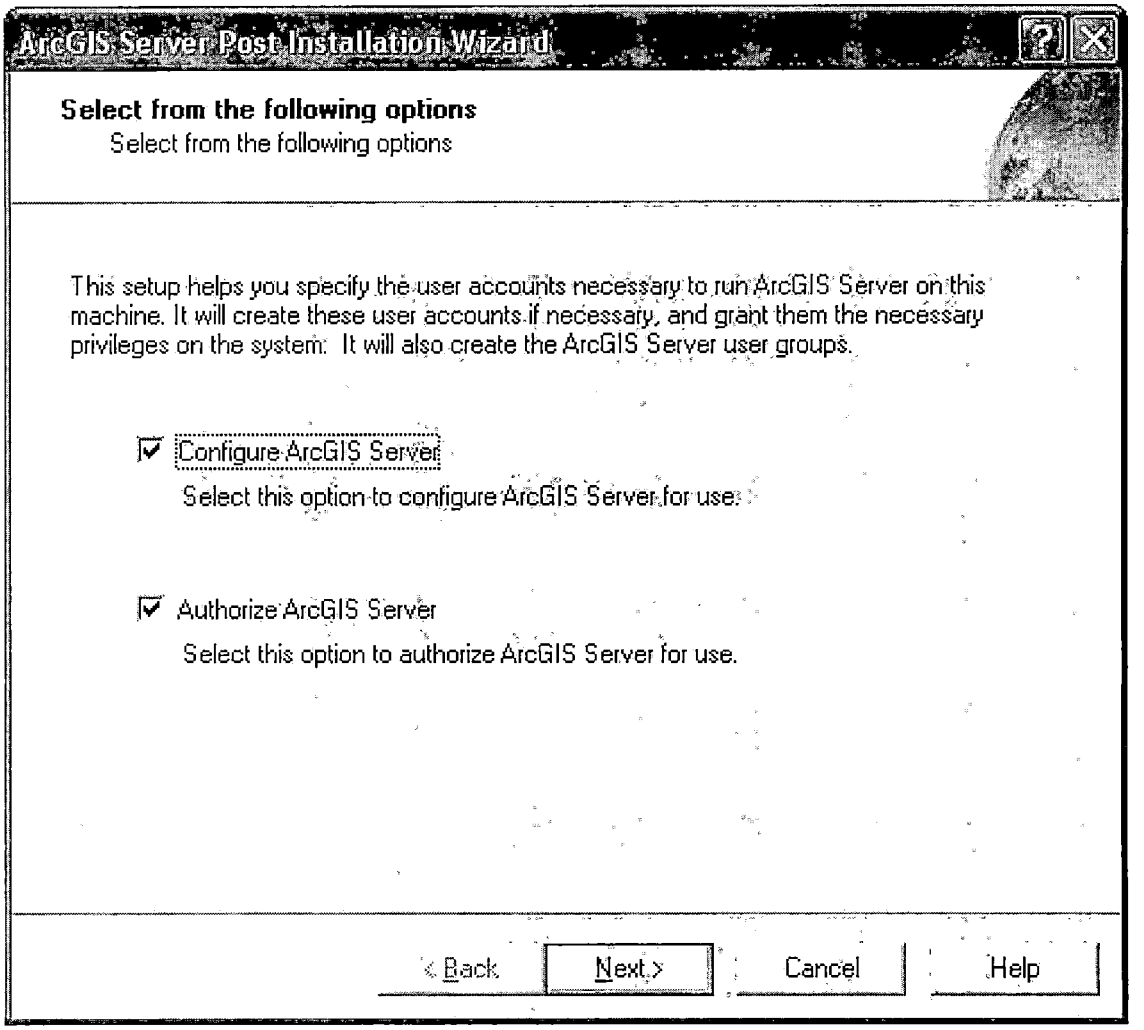


Figure 10. First Dialog in the Post-Server Installation

The second dialog asks for two account names, one for Server and the other for Container respectively. Keep it

consistent. The figure below displays how this should look. The password chosen can be based on the Administrator's discretion.

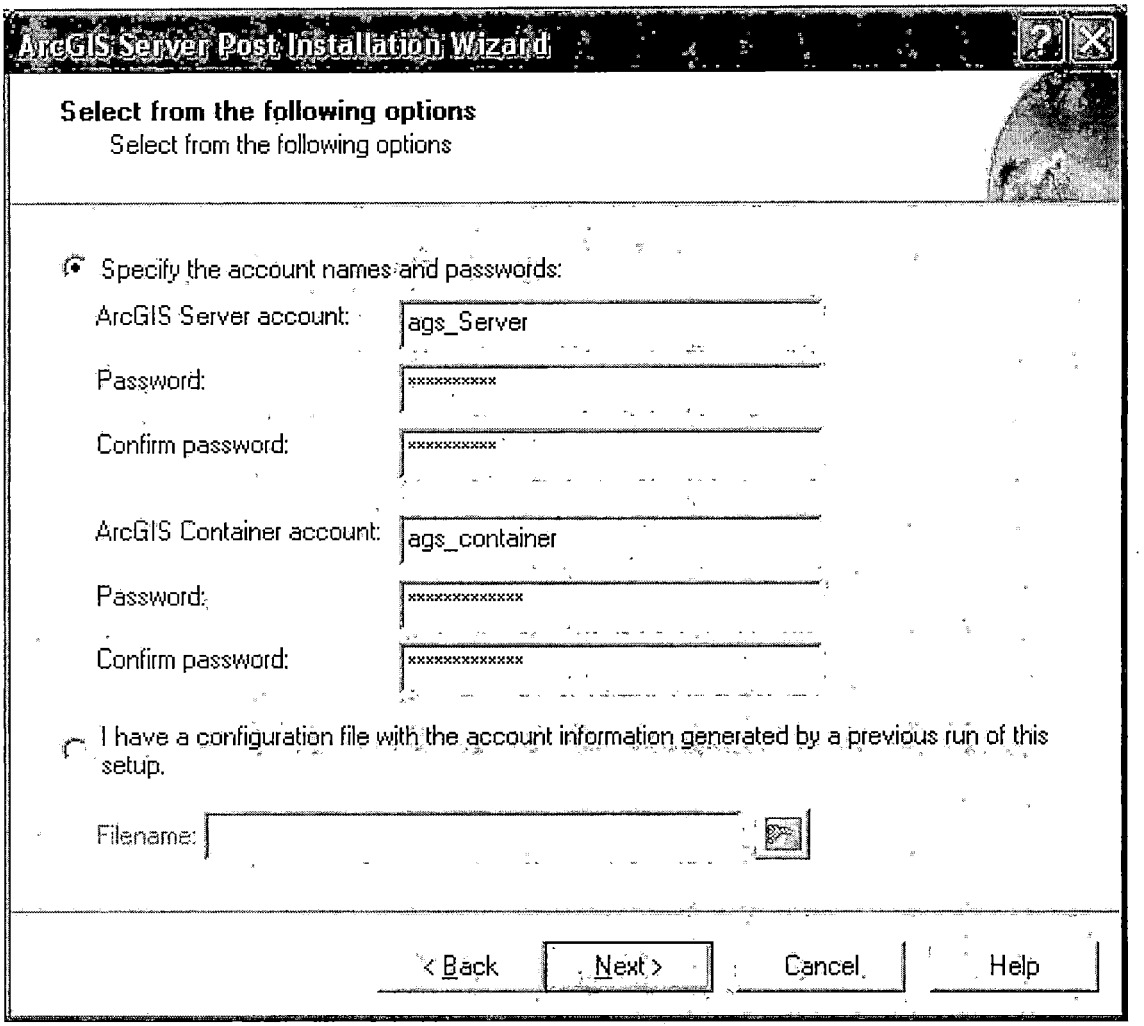


Figure 11. Second Dialog in the ArcGIS Server Post-Installation Process

The next dialog allows for specification of a proxy server. There is no proxy server used for HazWeb so the default as shown below is taken.

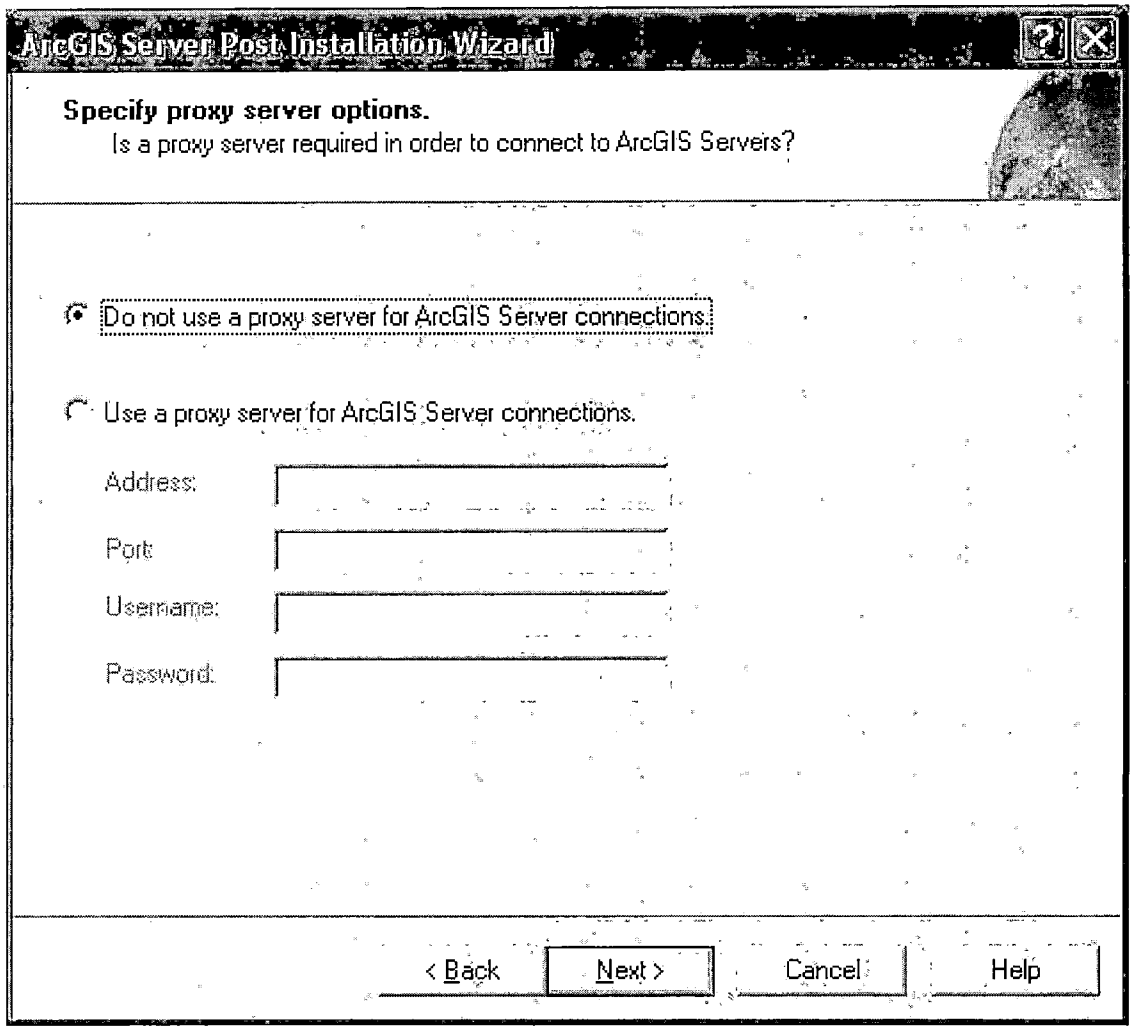


Figure 12. Third Dialog in the Post-Server Installation

Again, the default will be taken and no configuration file will be exported. This would be helpful if this process was re-occurring. If it was, this configuration file could be specified and used instead of having to manually specify all settings again.

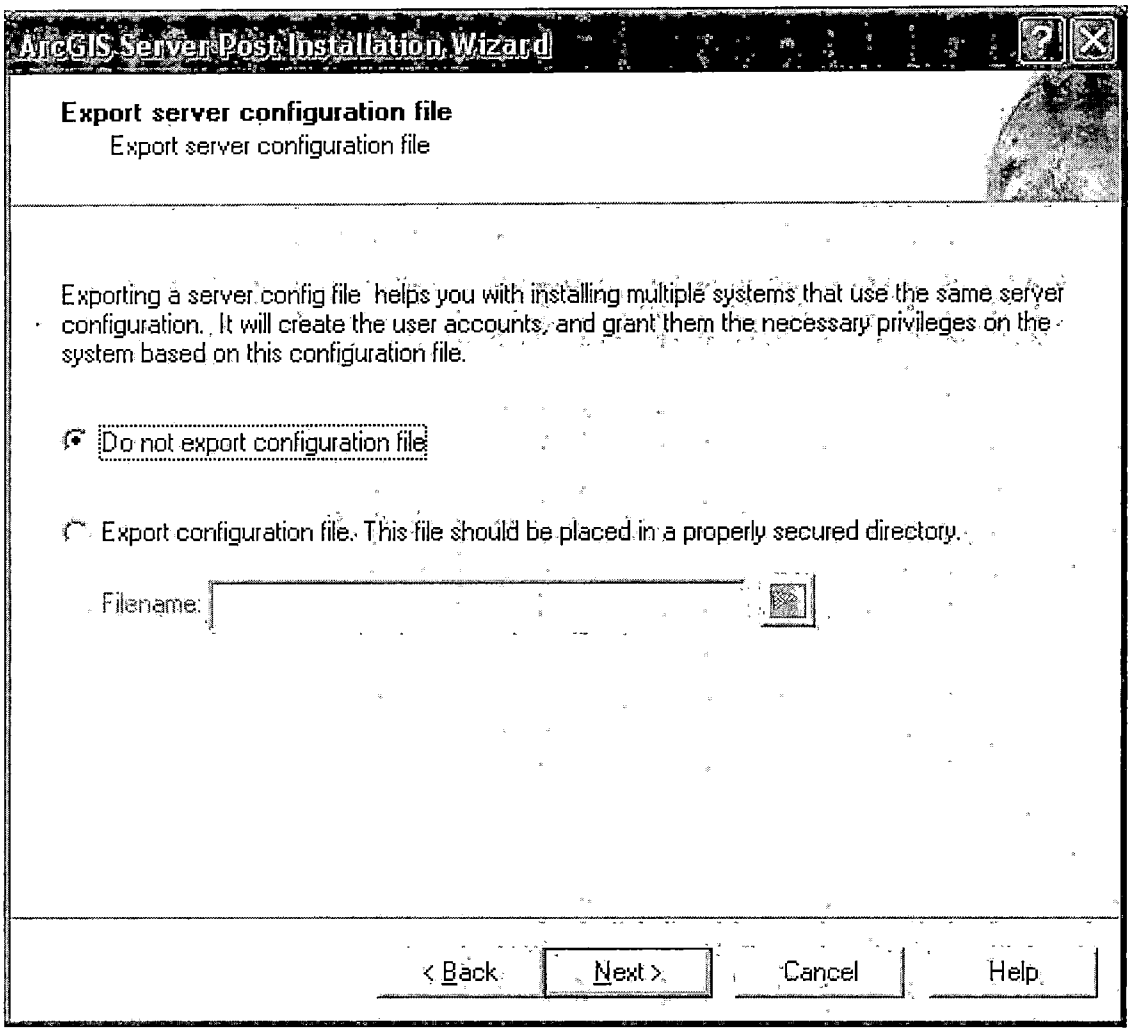


Figure 13. Fourth Dialog in the Post-Server Installation

The last dialog is a summary of the settings chosen. If everything looks OK, click "Install," otherwise, go back and fix incorrect settings.

4.3.3 Authorize ArcGIS Server for Use

After ArcGIS Server is configured, it needs to be authorized for use. All registered ArcGIS Server users have an .ecp, authorization, file. This file is a

standard text file that describes the product, version, and level of authorizing available. The figure below displays a dialog of the authorization wizard.

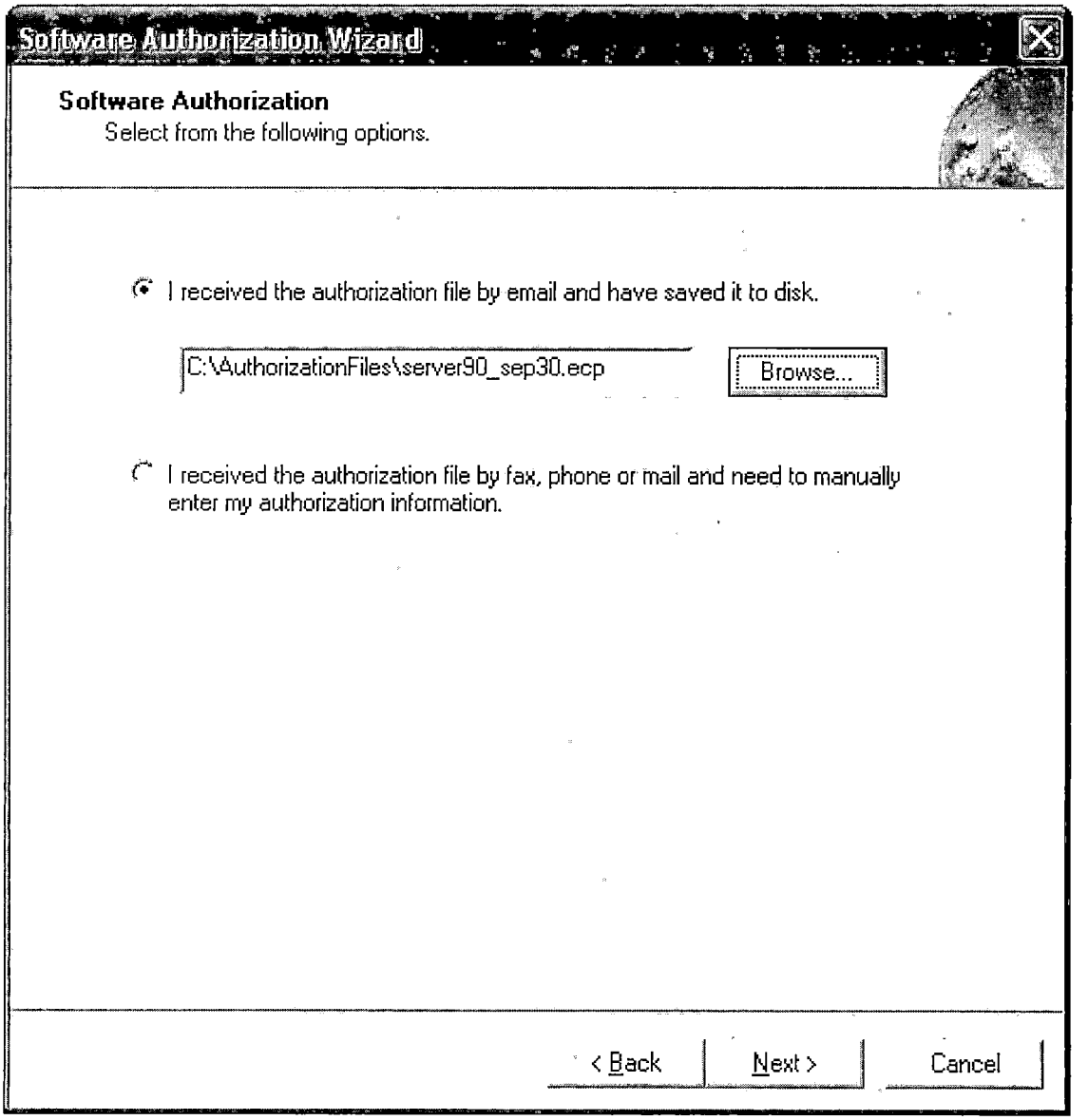


Figure 14. Software Authorization Wizard Used to Authorize ArcGIS Server

Each SOC machine must be authorized this way. Since HazWeb works on one machine, the authorization process need only happen once.

4.3.4 Install the .NET ADF

There is an additional installation for the Software Developer Kit, also known as the .NET ADF. This installation provides the framework to build Web applications that make use of ArcObjects running in the GIS server. It comes on a separate CD in the same media kit as the CD for ArcGIS Server. In order for this to install correctly, the .NET Framework version 1.1 must be installed on the system. If it is not, it can be downloaded free from Microsoft as a standard Microsoft Windows Update.

The steps are very similar as when installing ArcGIS Server.

1. Insert the ArcGIS Server CD into the CD drive to automatically launch the setup program.
2. During the installation you will be asked to read the license agreement and accept it, or exit if you don't agree with the terms.
3. The ArcGIS Server install allows you to select the features you would like to install.

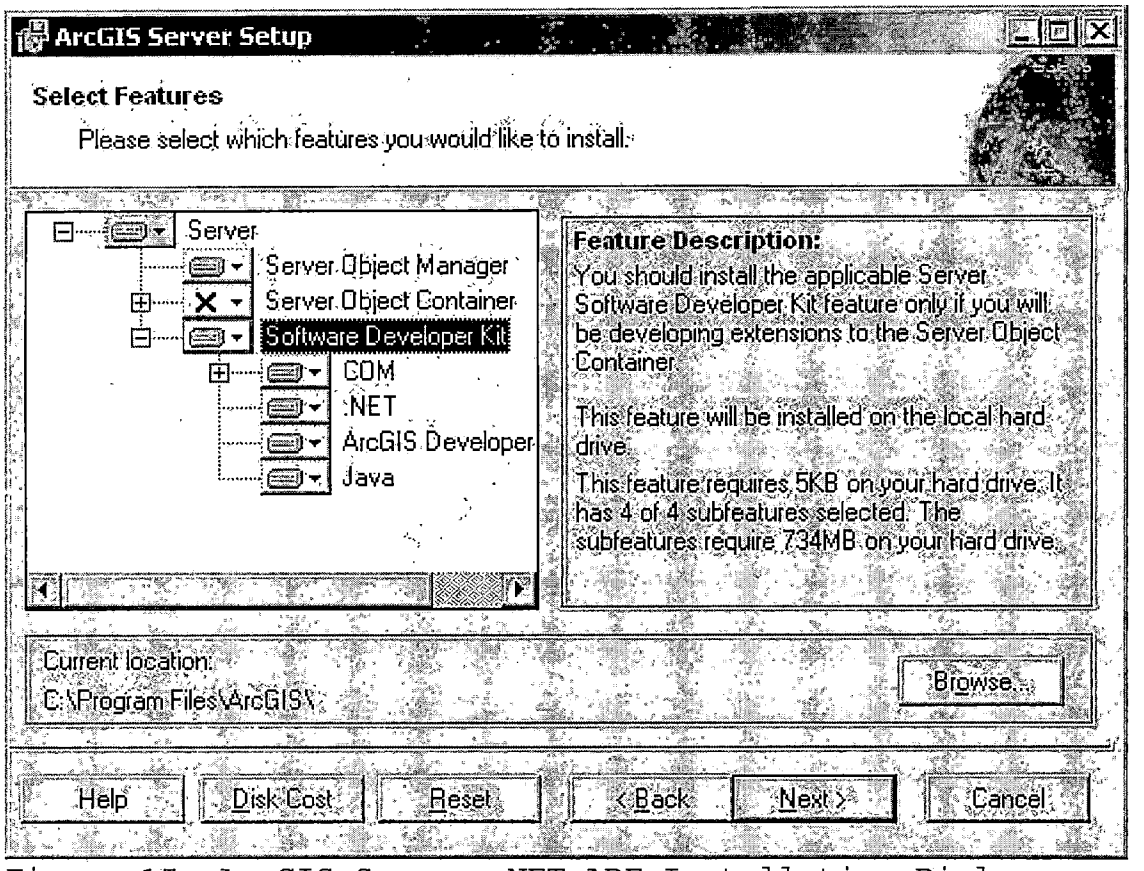


Figure 15. ArcGIS Server .NET ADF Installation Dialog

4.3.5 Add Users to Agsadmin and Agsusers Groups

After this is done, the Server and Container accounts are created. These two names will then appear as new users in the User Accounts setting in Windows Control Panel.

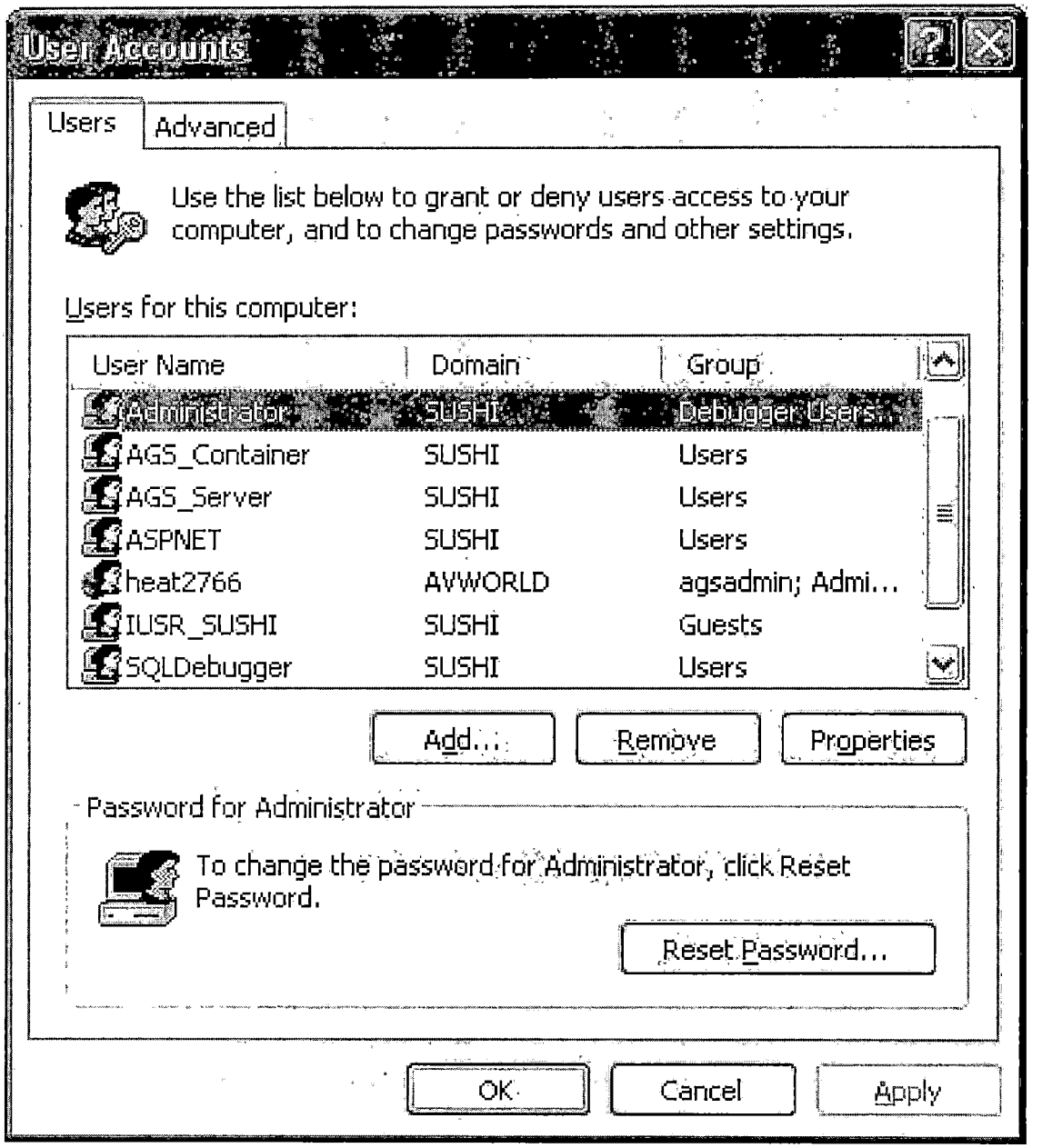


Figure 16. Windows XP User Accounts Dialog

In addition to the two user accounts created, two groups are also created. Agsadmin and agsusers are two groups created that allow either administrative privileges to ArcGIS Server or end user privileges. The

ArcGIS Server administrator would add the appropriate user login to either one or the other. The administrator would want to make certain that they have added themselves to the agsadmin group. If they were not added to this group, they would not be able to connect to the server via ArcCatalog. Standard Windows operating system tools can be used to do this. To access this, click on Start > Control Panel > User Accounts > Click on the Advanced Tab > Advanced. The following dialog displays. Any adding or removing of users can be performed here.

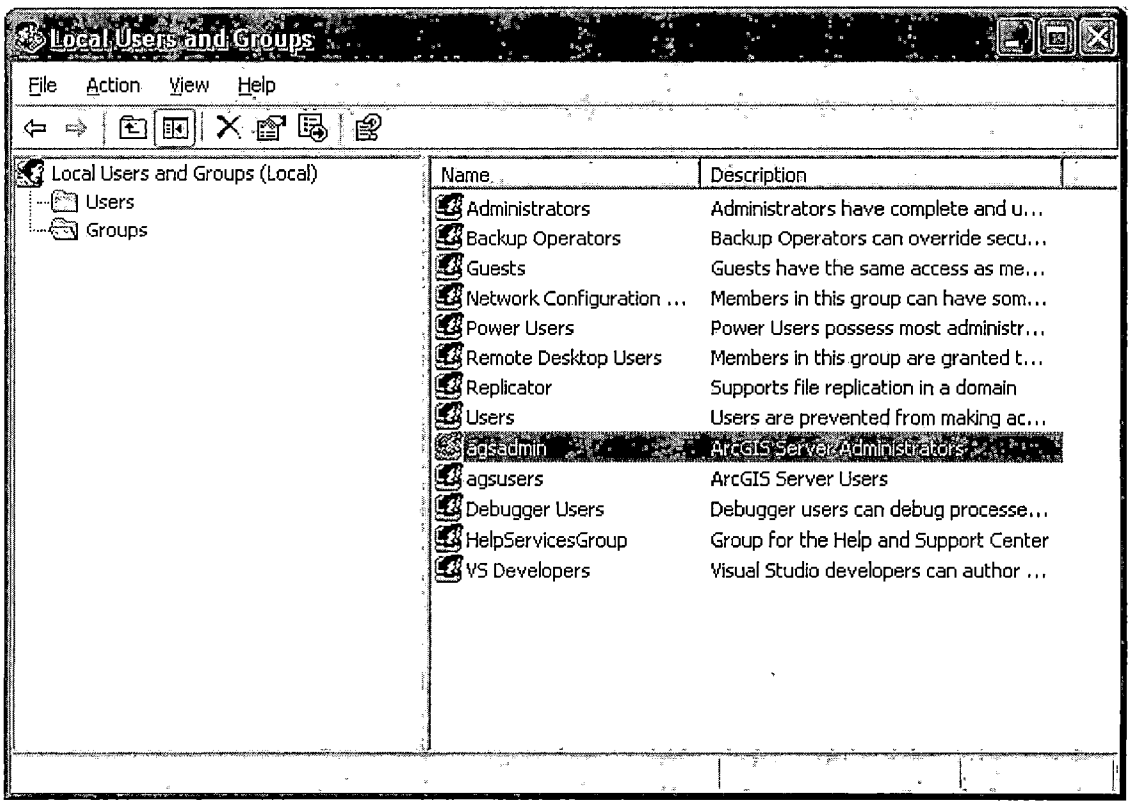


Figure 17. Local Users and Group Dialog

Before any of these changes take place, the administrator should log off and back on.

4.3.6 Grant Container Account Permissions

In order for the data hosted in the Server objects to be accessed and displayed correctly, the Container account must have at least read-access to the directories housing it. Again, this can be done using Windows Operating System tools. Open Windows Explorer and browse to the directory holding the data. HazWeb's data consists of personal geodatabases, shapefiles, an .mxd file, and four separate .loc files. Right-click on the folders holding this data > Properties > click on the Security tab > Add > type in the name of the Container account > OK. Make sure that this account has at least read privileges. The figure below displays this.

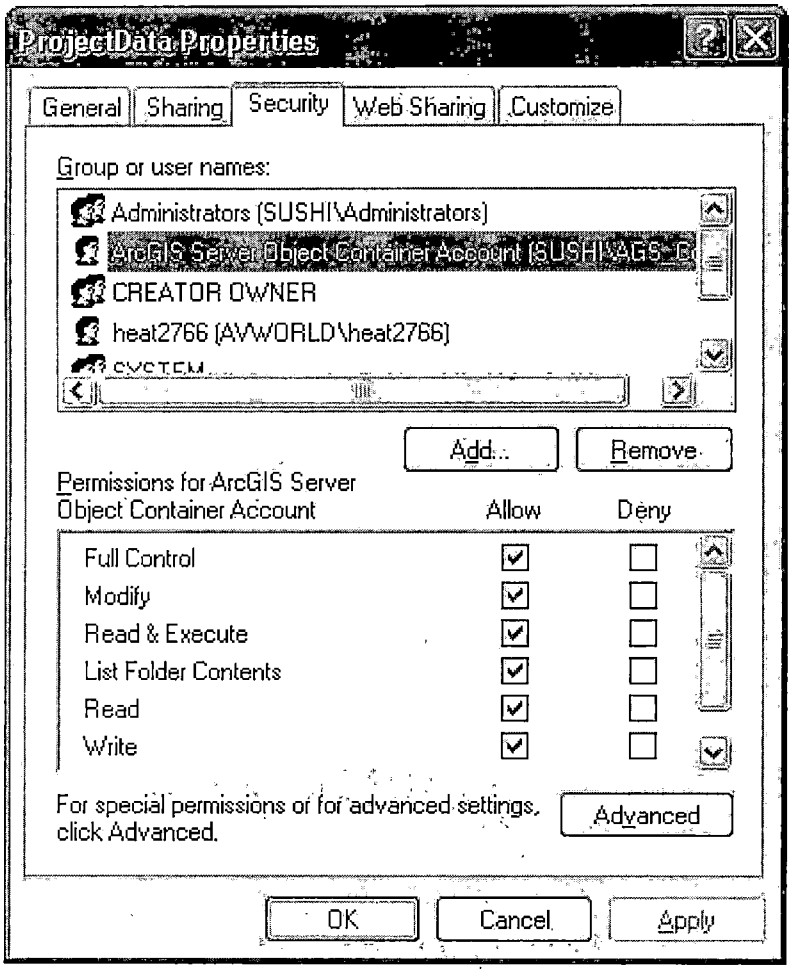


Figure 18. Giving the Container Account Access to the Data Hosted in the Server Objects

4.3.7 Create a Virtual Directory

Last, since a virtual directory is used in HazWeb, one must be created or else use an existing one. This can be done using standard Windows Operating System tools. The Container account must have both read and write permission to the folder specified as the virtual directory. If no virtual directory is created, it can be

done using standard Microsoft Windows Operating Tools. The steps below should be taken to set up a virtual directory:

1. Click on Start > Programs > Control Panel > Administrative Tools > Internet Information Services.
2. Double-click on the machine name > Web Sites > Right-click Default Web Site > New > Virtual Directory.
3. In the "Welcome to the Virtual Directory Creation Wizard," click "Next."
4. Give the Alias name for this directory. This is the name that will appear when you specify a location, e.g., http://sushi/HazWebProject
5. Browse to the location of the physical file on disk.
6. Click on at least the Read and Write check boxes.

   Note: The Default Web Site in Step 2 points to C:\\InetPub\wwwroot on disk.

A new virtual directory is created, the Container account needs to have both Read and Write permissions to this folder on disk. Again, using standard Windows System

Operating tools, open Windows Explorer, right-click on the folder representing the virtual directory > Properties > Security > Add the Container Account to this folder with appropriate permissions.

4.3.8 Adding the ArcGIS Server to ArcCatalog

Before any Server objects can be added, the Server must be added in ArcCatalog. This cannot be done unless the appropriate person trying to add it is added into the agsadmin group.

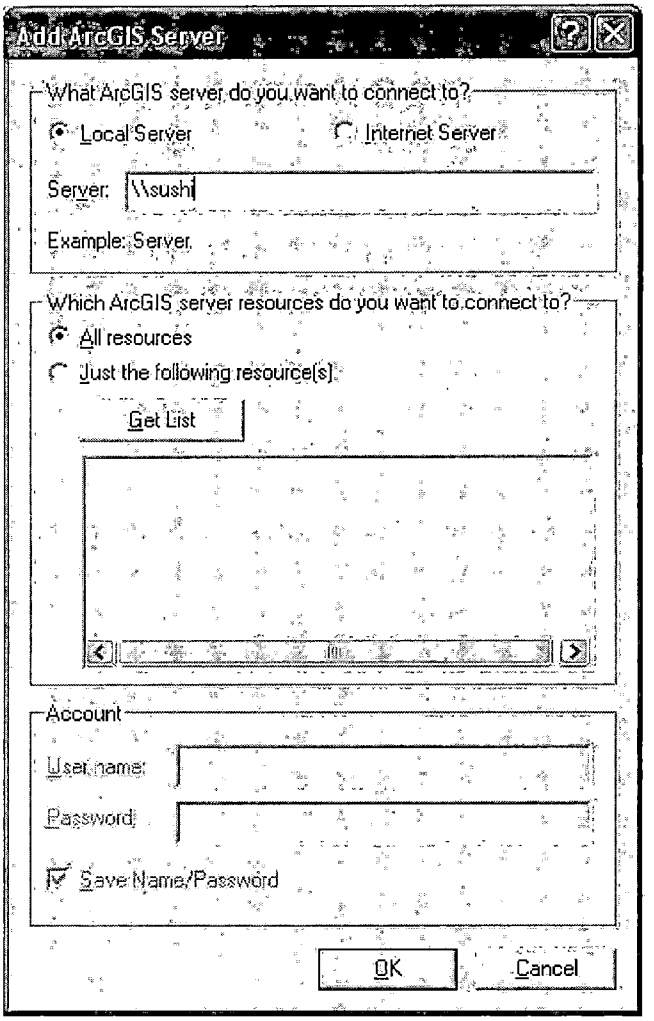


Figure 19. The Add ArcGIS Server Dialog in ArcCatalog

After the Server is added into ArcCatalog, the Server Properties need to be set up. This is where connections to the SOC machines are managed. For example, host names are specified, as are timeout properties, virtual directories, and logging properties.

There is a "Hosts" tab which all SOC's accessing the SOM machines are added. It is the SOC machine that does

the brunt of work, so it is important that these are added as needed. Since HazWeb is configured on one machine, the only name needed here is the name of the machine currently being used.

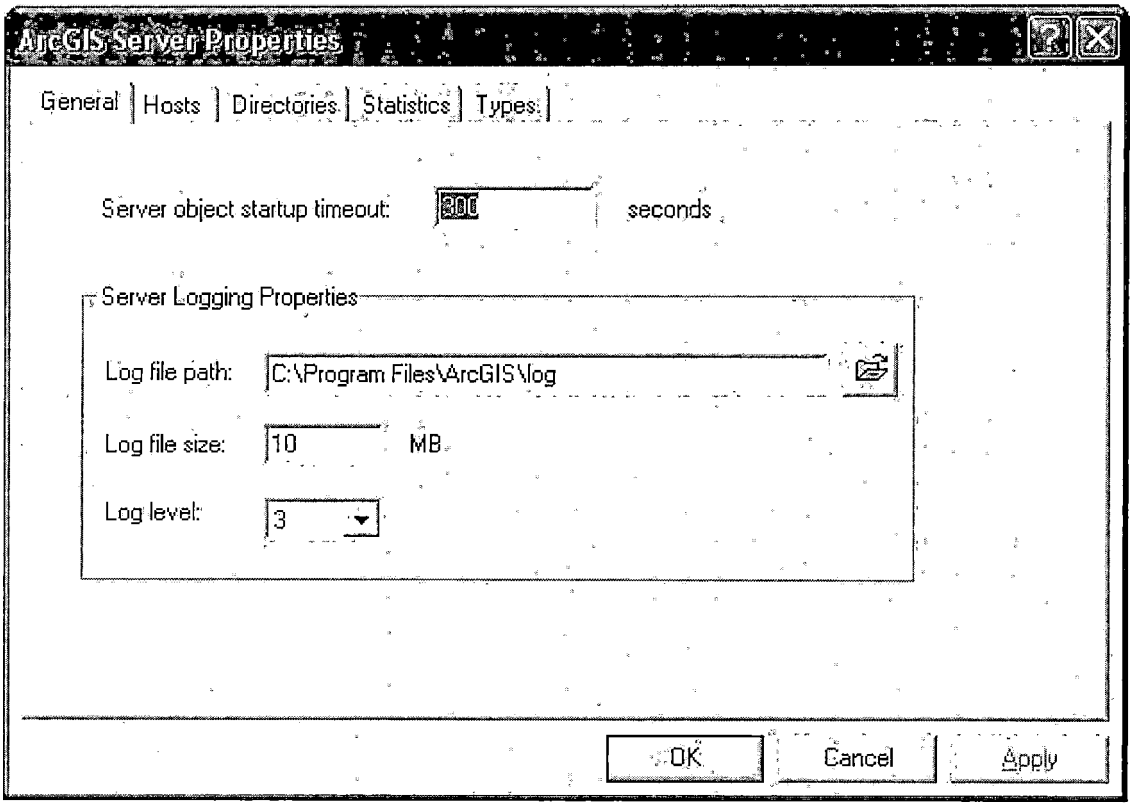


Figure 20. The ArcGIS Server Properties Dialog

4.5 ArcGIS Server Object Configuration

One MapServer and four GeocodeServer objects were needed for this project. The MapServer object is what is used to display an already-existing .mxd file. The actual MapServer object gets created in ArcCatalog. The

following figure displays what the .mxd file looks like in its authoring program, ArcMap.

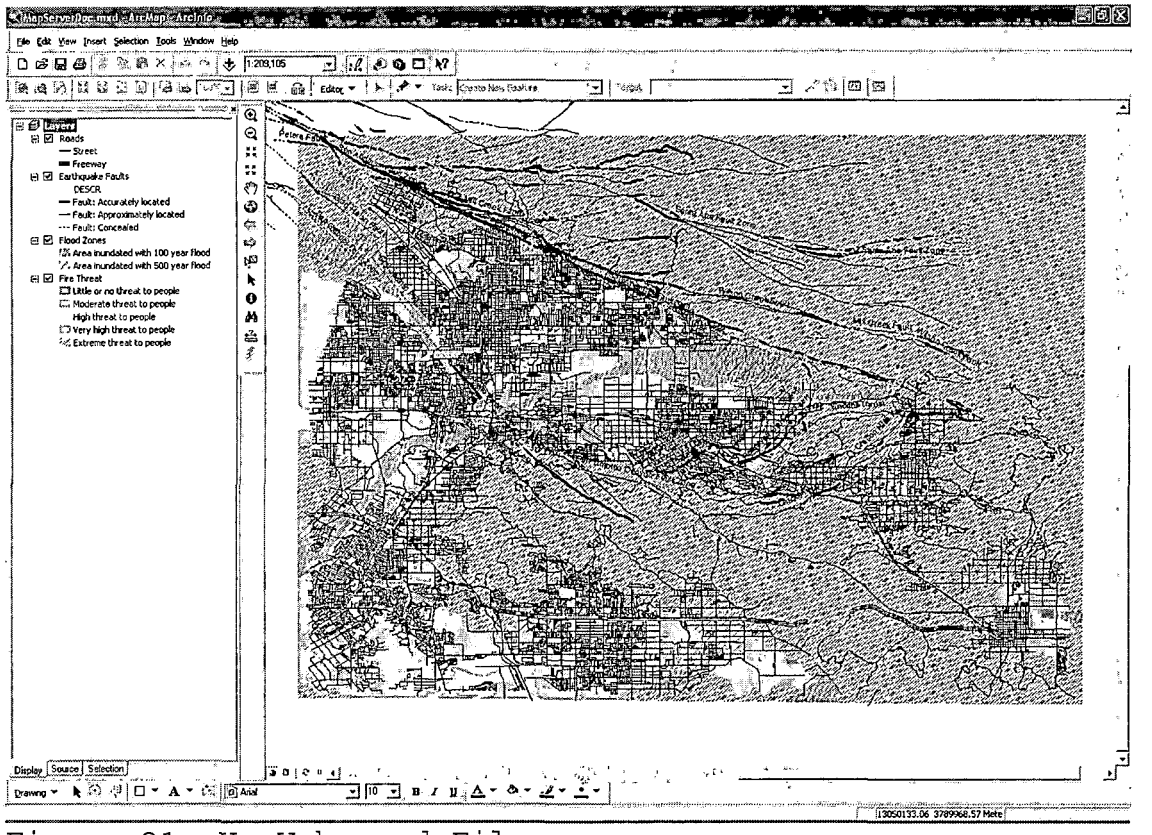


Figure 21. HazWeb .mxd File

After the .mxd is authored, a MapServer object is created. The figure below shows the steps taken when creating this MapServer object.

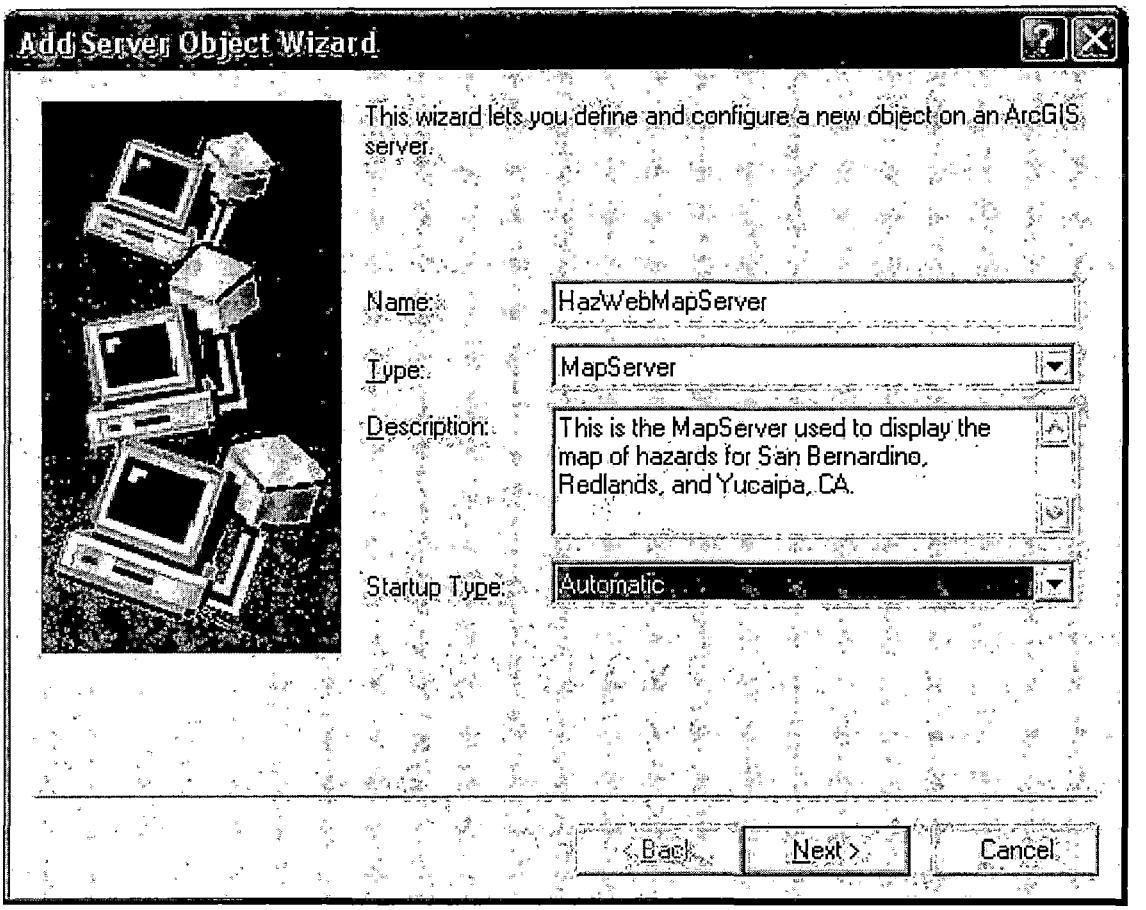


Figure 22. First Dialog in Add Server Object Wizard

This screenshot shows the name given to the MapServer objects as, HazWebMapServer, the type, description, and startup type. The startup type states whether the object should be made available as soon as ArcCatalog is opened. By default, it is set to automatic. It can also be started manually each time ArcCatalog is opened.

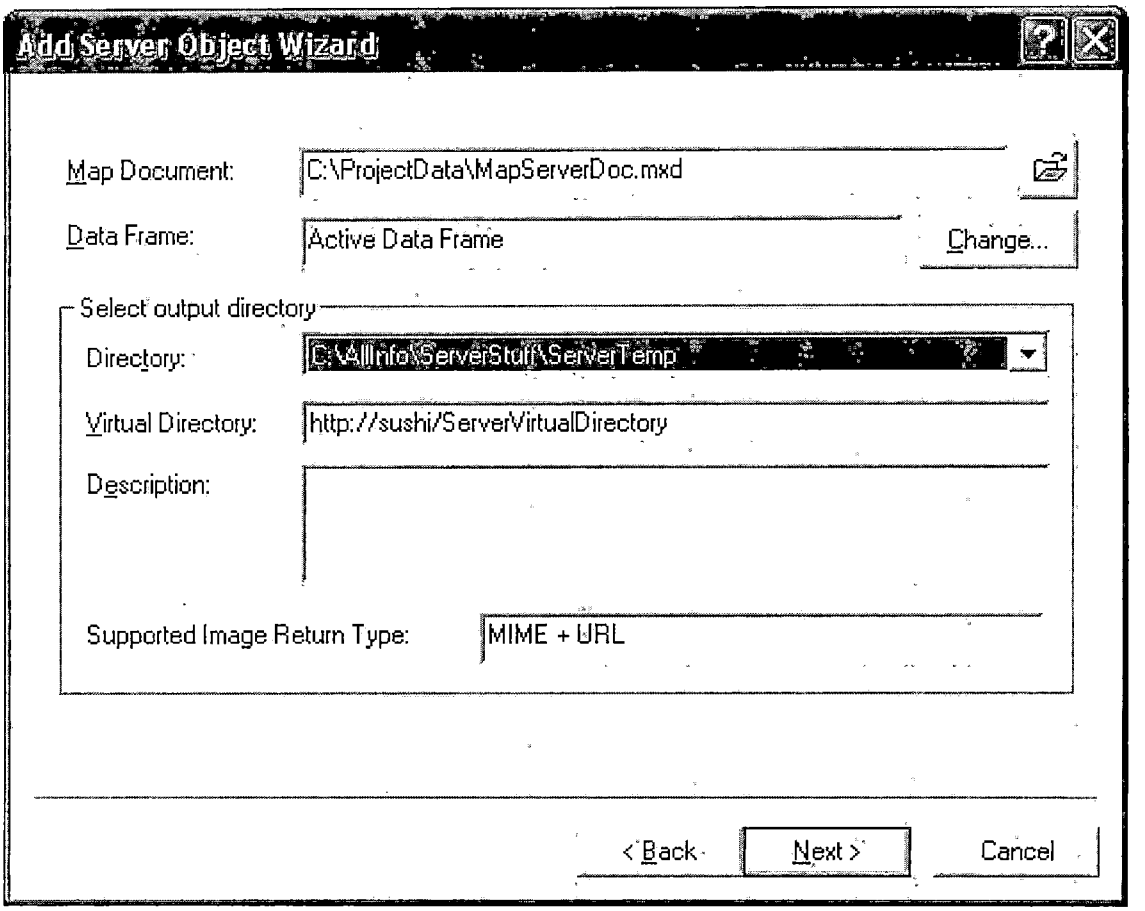


Figure 23. Second Dialog in the Add Server Object Wizard

This second dialog is where the actual .mxd file is specified. An .mxd can contain more than one map. These maps are called data frames. The data frame needed for the MapServer object should be given. A virtual directory is provided. This is specified when configuring the server properties. A virtual directory allows a file system directory access via a URL. It is the means of publishing data over the Internet. Images for the

MapServer object will get written out to the virtual directory.

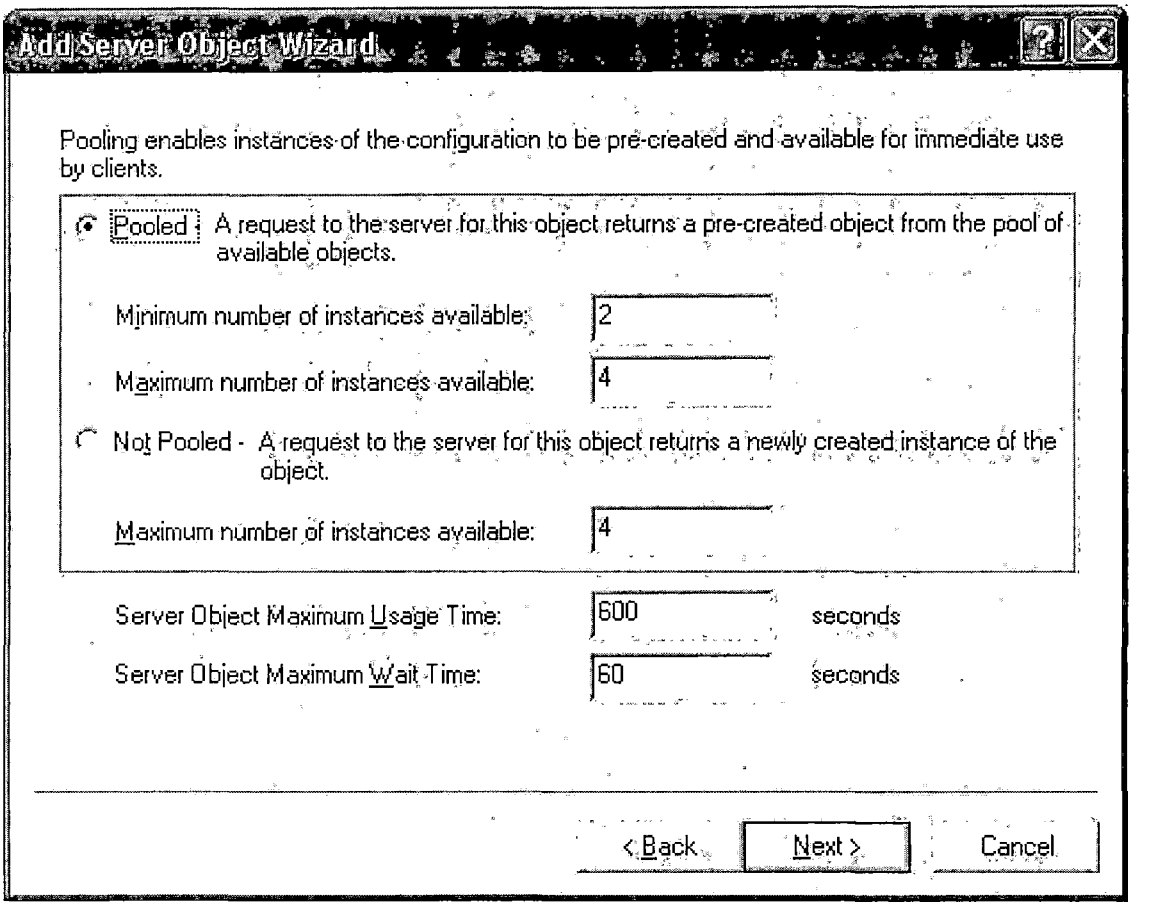


Figure 24. Third Dialog of the Add Server Object Wizard

The dialog above allows a choice between pooled and non-pooled server objects. When a server object is pooled, a single instance of a server object can be shared between multiple client application sessions. When the server object is started, the GIS server creates the minimum number of server objects specified. When a client

application requests a server object, the SOM passes it a reference to one of the server object instances. The client application keeps that reference for the duration of the request, and then releases it back to the server.

Non-pooled server objects are instances of a server object created for one client application's use. It is destroyed when released by the application to the server. The maximum number of non-pooled server objects that can be created are limited to what is set in the above configuration. If more requests come in than the maximum number of instances allotted, they will sit in a queue until the number of users drop below the maximum.

Usage time indicates the maximum time a client can hold on to an object. This is to ensure the client application correctly releases the server objects. Wait time is the maximum time a request can wait in the queue for a server object request before the request is timed out (Laframboise & Markham, p. 2-36).

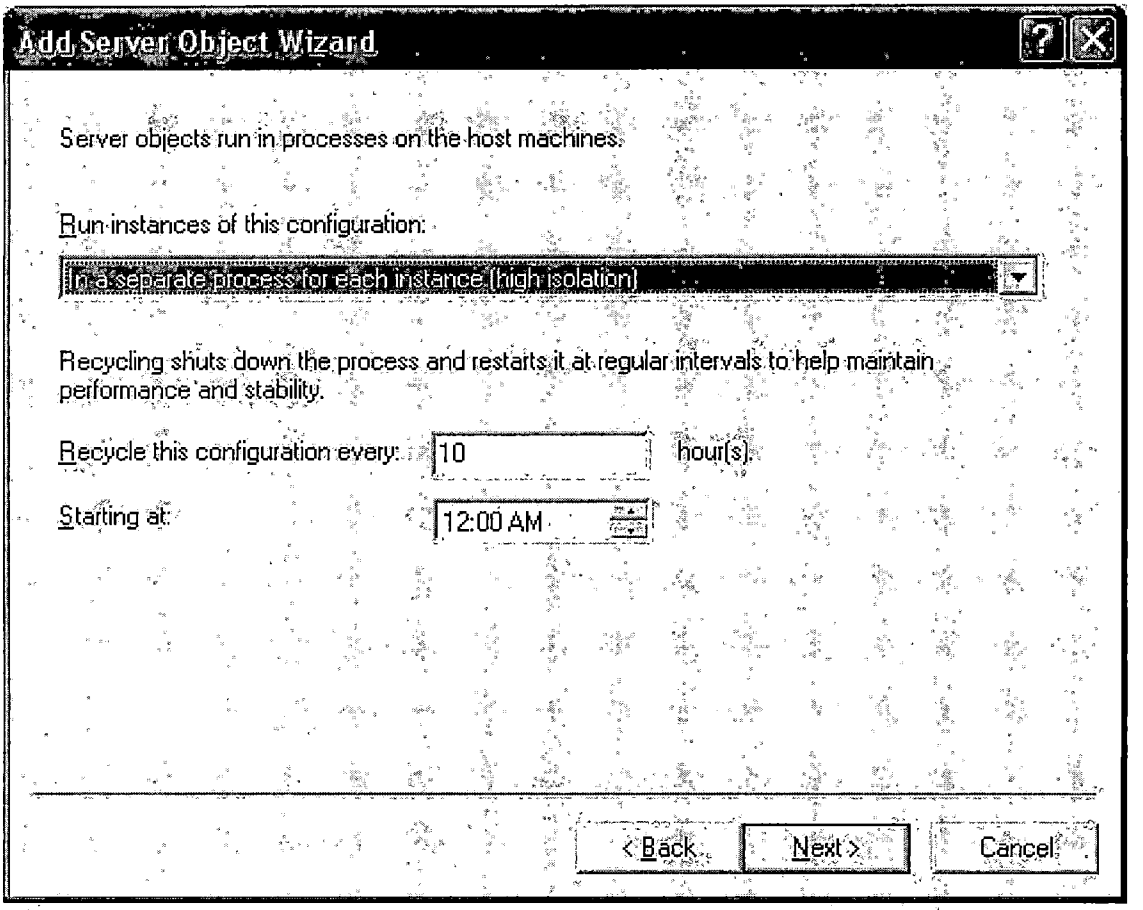


Figure 25. Fourth Dialog of the Add Server Object Wizard

Server objects run within processes on the SOC machines. They can be configured to either high or low isolation. High isolation indicates that each server object instance will run within a separate process space. An advantage to this is robustness, since these objects do not share processes. If an error occurs on one of these objects that cause it to crash and shut down, it will not affect other server objects.

Low isolation indicates that more than one server object instance can run within the same process space. A maximum of four instances can be run in one process. An advantage of low isolation is that it can free up server resources.

Recycling allows server objects that have become unusable to be destroyed and replaced with new server objects. Through reuse, a number of things can happen to a server object to make it unavailable for use by applications. For example, an application may incorrectly modify a server object's state making it unavailable to other applications or sessions (Laframboise & Markham, p. 2-37).

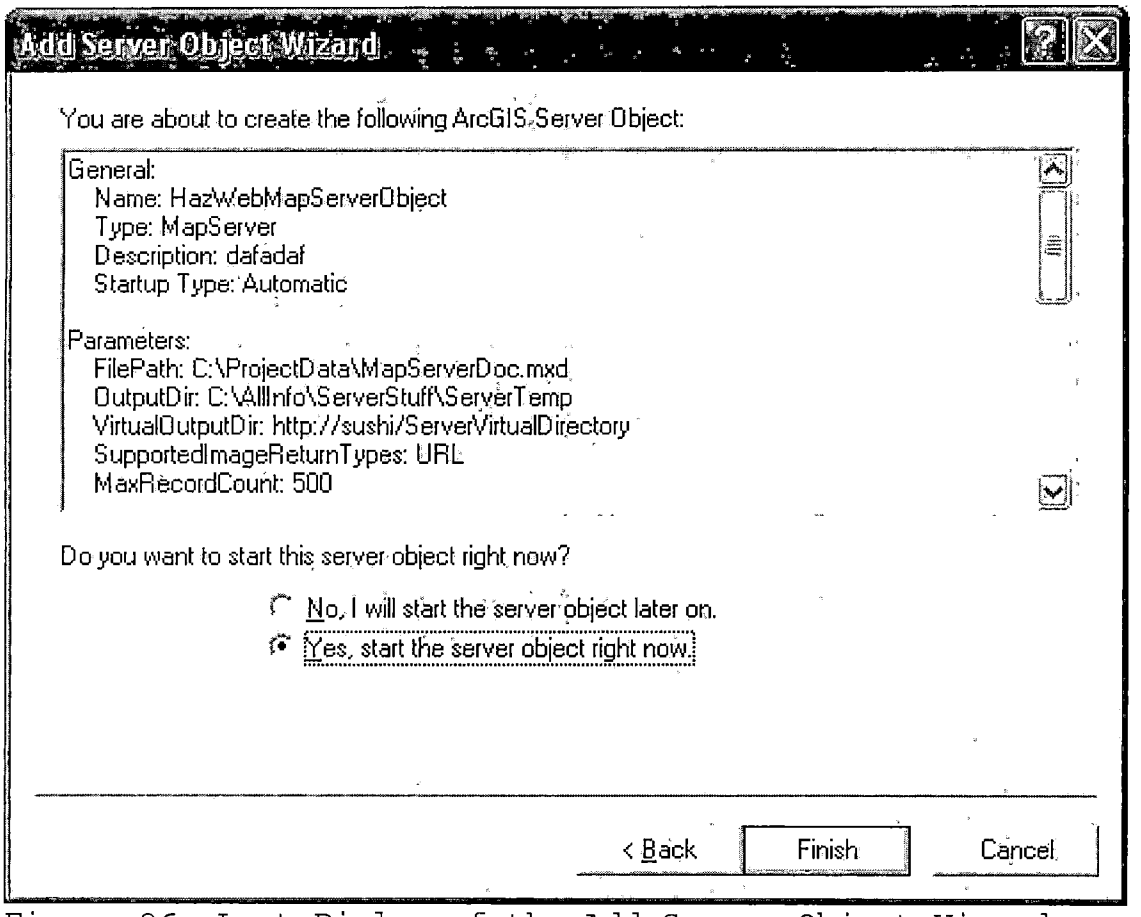


Figure 26. Last Dialog of the Add Server Object Wizard

This last dialog above gives a general description of the Server object about to be created. In addition to this, the object can be started as soon as "Finish" is clicked or until the user explicitly starts it within ArcCatalog.

The figure below shows how the MapServer object appears in ArcCatalog after being created by the above steps.

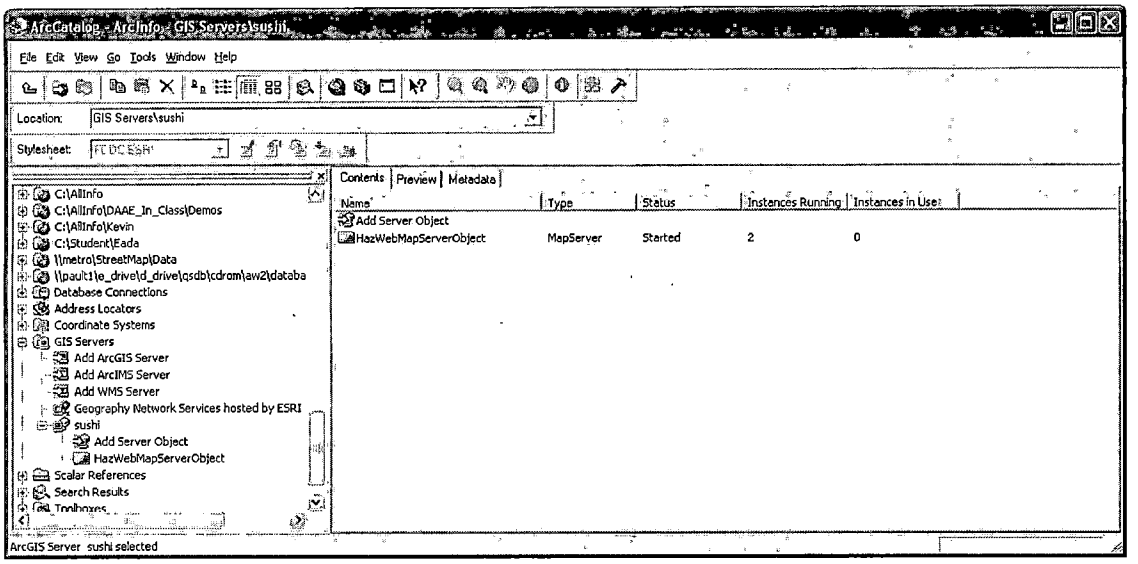


Figure 27. HazWebMapServerObject Shown in ArcCatalog

The last five figures shows how the MapServer object was created. Four Geocode server objects were created in the same fashion. Before a Geocode server object can be created, a .loc file needs to be authored by ArcCatalog. The .loc file is generated based off of a style. The style is a template on which the address locator is built. Each template is designed to accommodate a specific format of address. This project uses four styles, consequently four separate Geocode objects. Two of the styles are "Streets with Zone," one is "Zip code," and the last is "US Cities with State." The zone specified with the streets for the first two styles is a zip code and city.

The web application allows the user to type in an address as either: street address plus a zip code, street address plus city, only a zip code, or only a city. Based on the input values from the user, the appropriate Geocode object and its respective style are used. Regardless of the type of style used, the steps needed to create a Geocode object remain the same. The steps below illustrate how to create a GeocodeServer object.

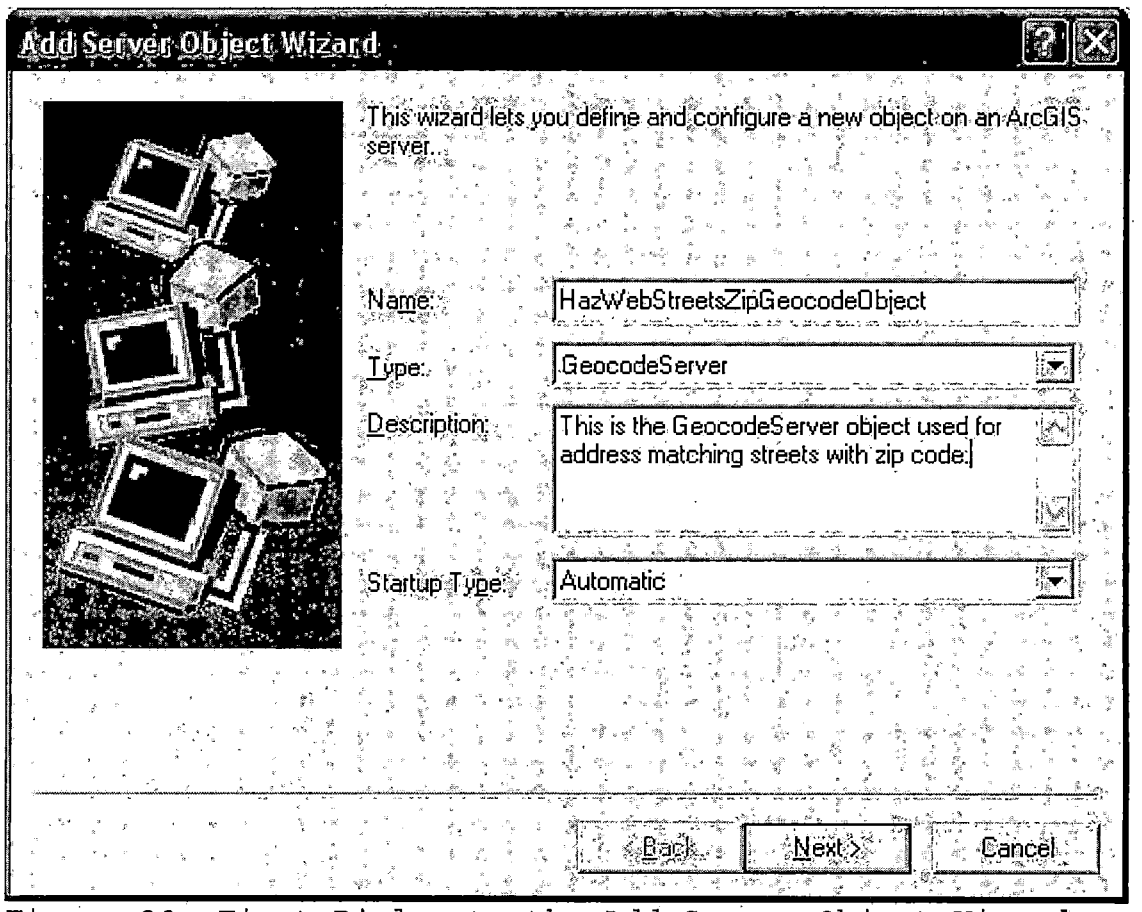


Figure 28. First Dialog to the Add Server Object Wizard for GeocodeServer Objects

Figure nine displays the first dialog when creating a GeocodeServer object. This dialog is identical to the MapServer object except instead of a MapServer, it is a GeocodeServer type.

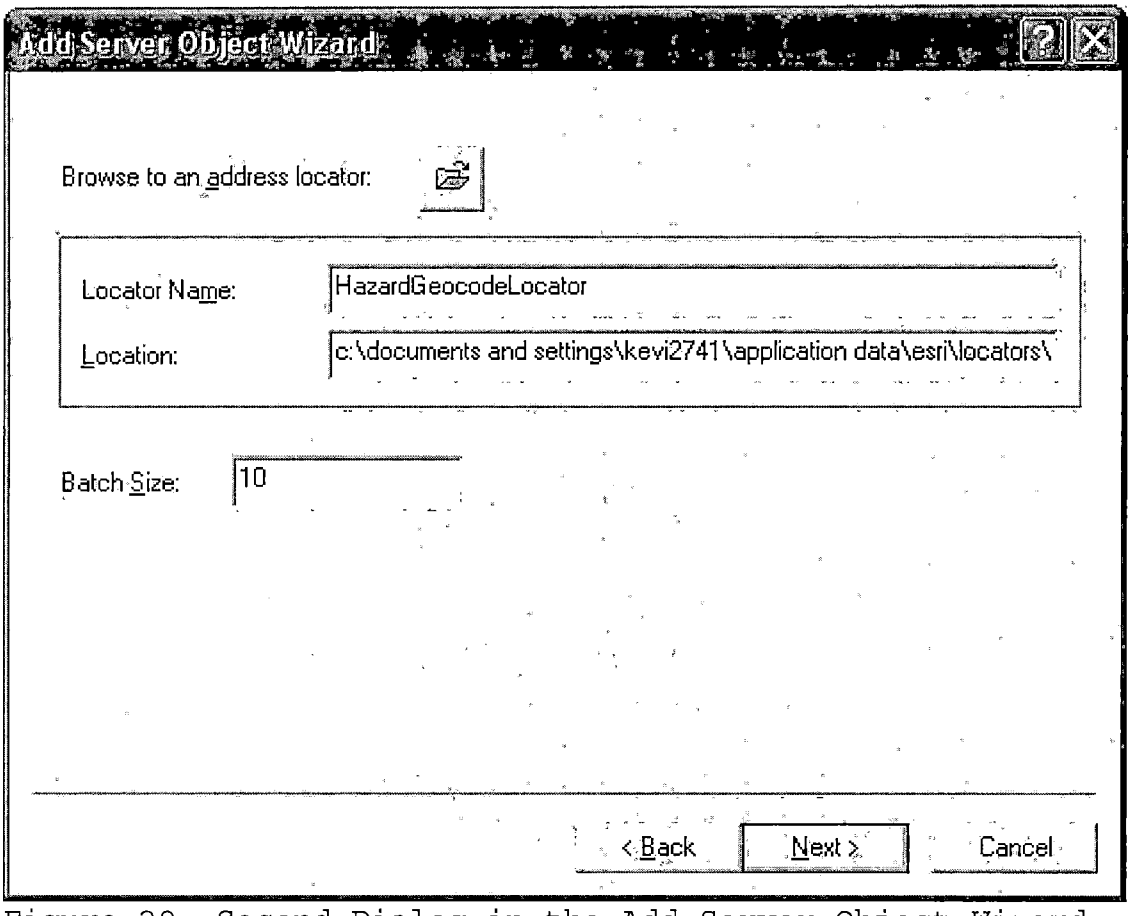


Figure 29. Second Dialog in the Add Server Object Wizard for a GeocodeServer Object

Figure 10 shows the second dialog when creating a GeocodeServer object. The .loc file is specified as is the name and its location. The batch size is the size

specified for return values when doing a batch address match. This parameter does not pertain to this project since the project only takes in single address values one at a time.

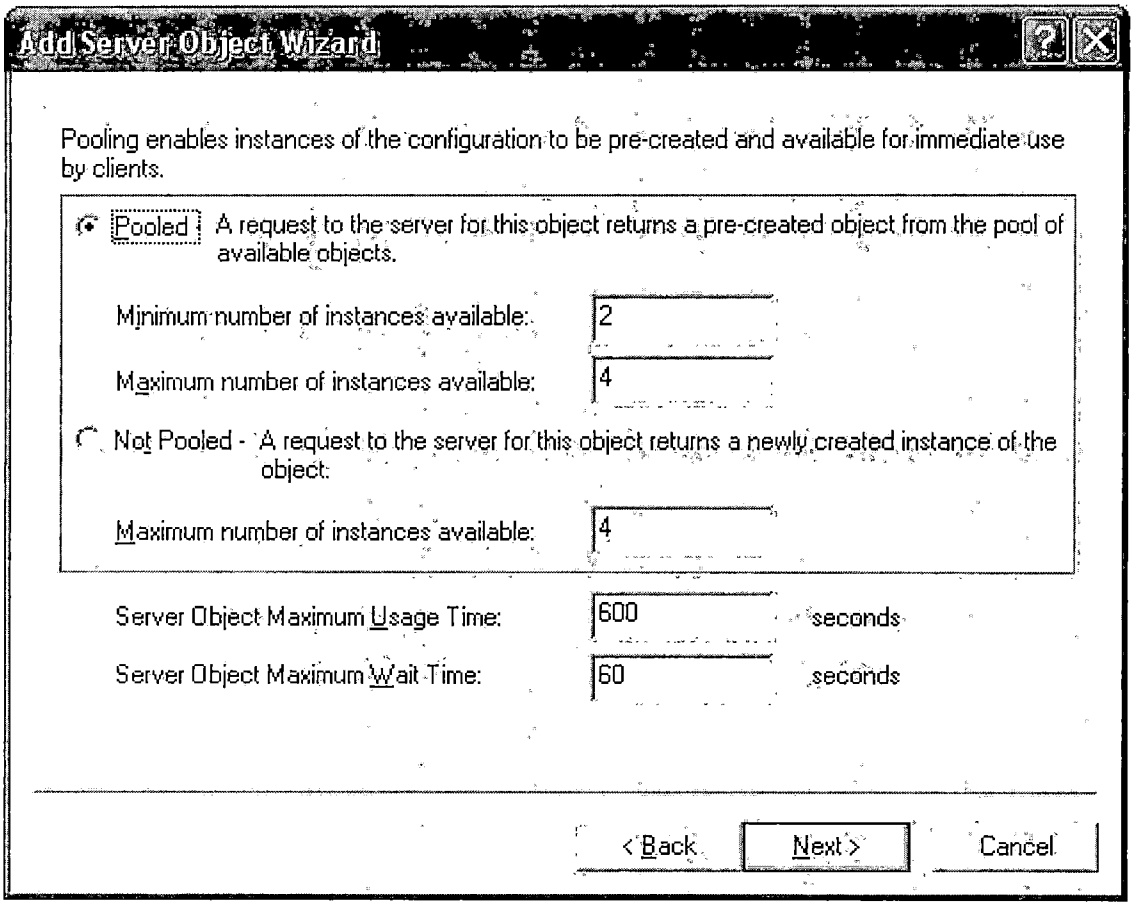


Figure 30. Third Dialog in the Add Server Object Wizard for GeocodeServer Objects

The figure above allows the specification of pooled or non-pooled objects. The same premise applies to

pooling and GeocodeServer objects as they do to MapServer objects.

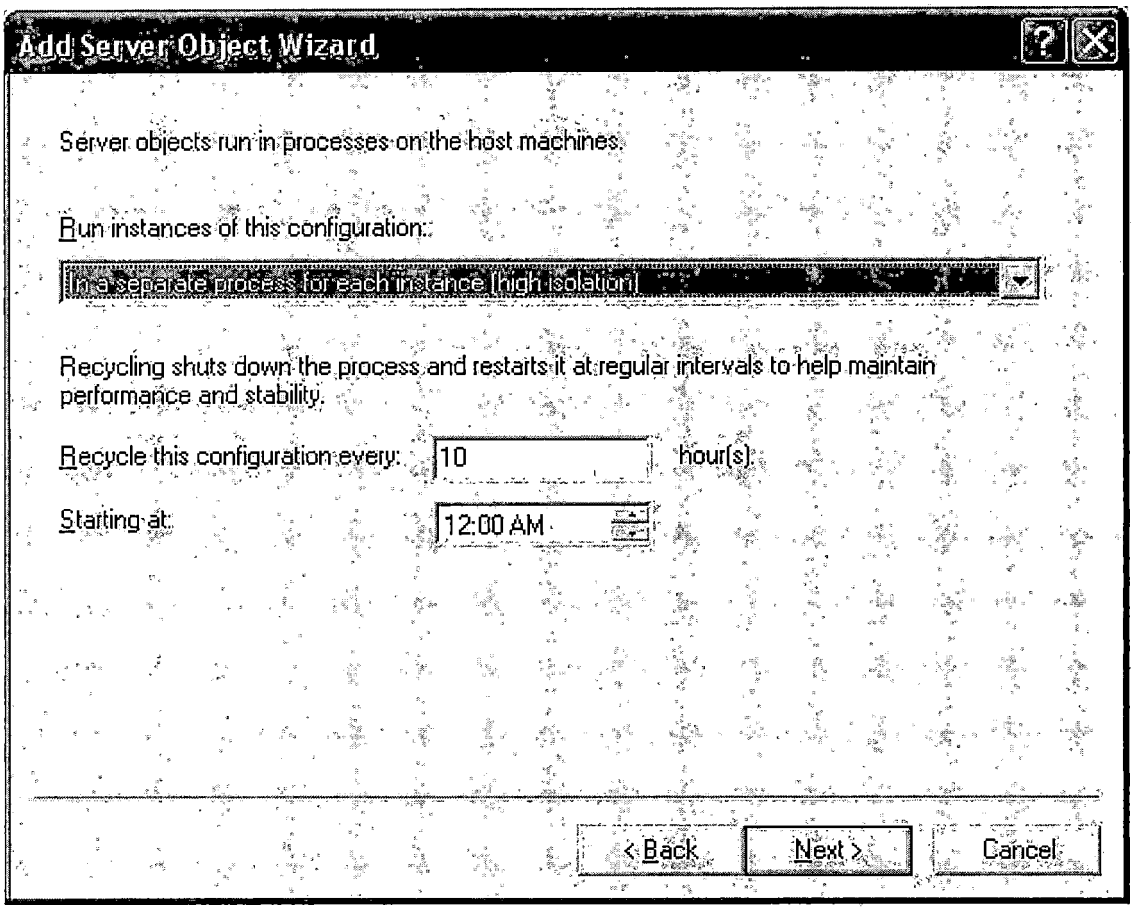


Figure 31. Fourth Dialog of the Add Server Object Wizard for GeocodeServer Objects

The figure above allows the specification of high or low isolation for the GeocodeServer object. The premise regarding high and low isolation for MapServer objects also applies for GeocodeServer objects.

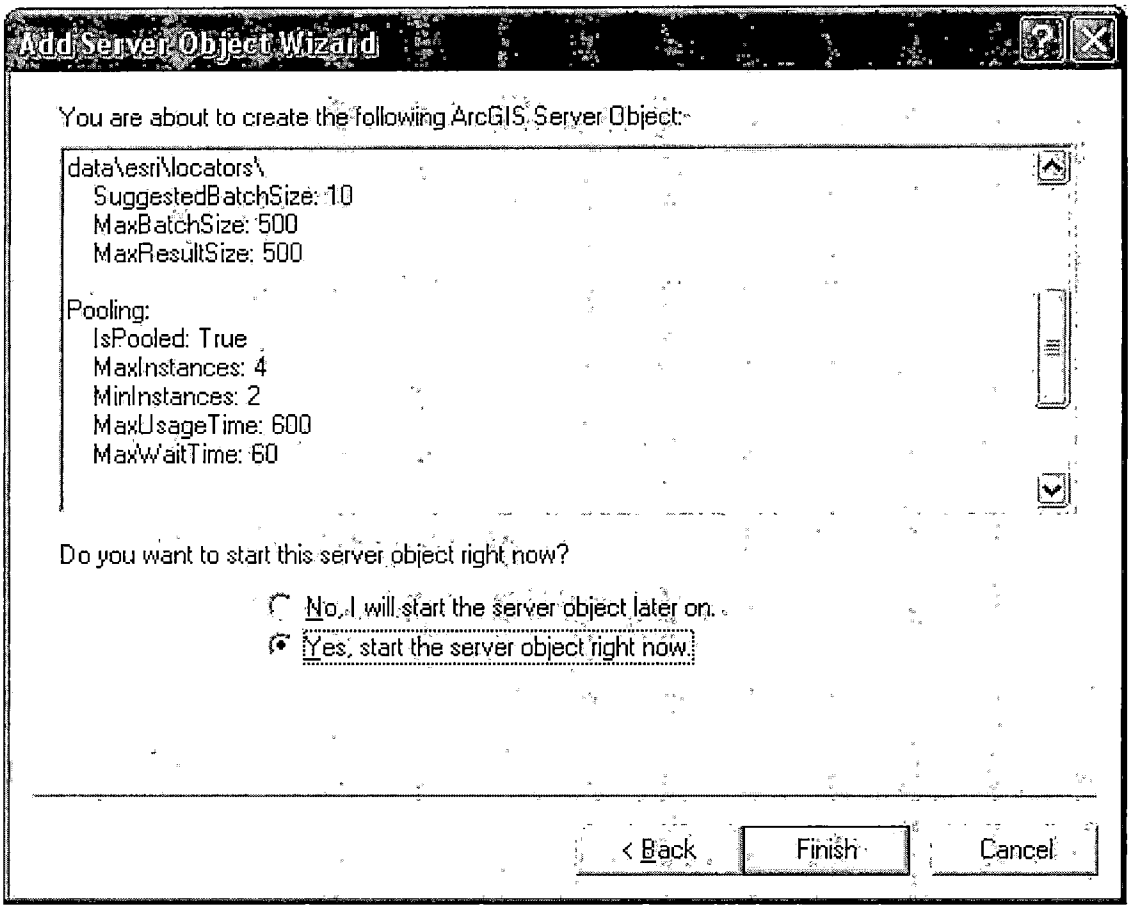


Figure 32. Last Dialog for the Add Server Object Wizard for GeocodeServer Objects

As with the MapServer object, the GeocodeServer object can be started automatically upon clicking "Finish", or only by manually starting it via ArcCatalog by the Server administrator. The resulting GeocodeServer object displays as the MapServer object in ArcCatalog. Figure 14 below illustrates this.

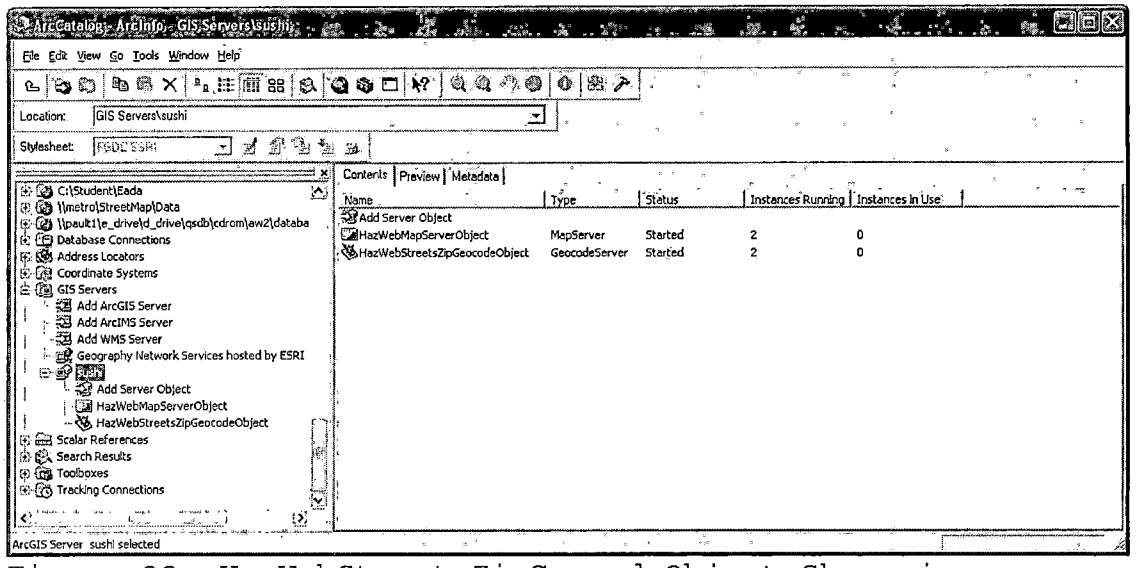


Figure 33. HazWebStreetsZipGeocodeObject Shown in ArcCatalog

4.5.1 Data and Application Backup

Since there is such an abundance of spatial data used in HazWeb, it is crucial to have backups in case of possible corruption. There is no editing performed on the data, but in the event of a possible operating system malfunction, it is a good idea to have a copy of the data stored somewhere else. This is easy to do considering that most of the data used in HazWeb is in Microsoft Access format. The other data, such as shapefile, .mxd, and .loc files, can easily be backed up using standard Windows Operating System tools to another machine across the Local Area Network or offline.

HazWeb is an ASP.NET application, the files needed to run this application are easy to copy and store on another machine, similar to the data backup. The files comprised in this web application are: AddGraphicElement.cs, AssemblyInfo.cs, Buffer.cs, Default.aspx, Default.aspx.cs, Default.aspx.resx, ErrorPage.aspx, ErrorPage.aspx.cs, ErrorPage.aspx.resx, Functions.cs, Global.aspx, Global.aspx.cs, Global.aspx.resx, HazWeb.csproj, HazWeb.csproj.webinfo, HazWeb.sln, Map.aspx, Map.aspx.cs, Map.aspx.resx, and HazWeb.css, and Help.htm

If the machine hosting HazWeb were to go down, this web application could be run from another machine as long as it was configured with the above-mentioned steps and the Solution (.sln) file pointed to the correct project location project. Also, the project directory must be configured as a web directory. The steps below show how this is done:

1. Click on Start > Programs > Control Panel > Administrative Tools > Internet Information Services
2. Double-click on the machine name > Web Sites > Default Web Site

3. Right-click on the folder containing the project > Properties

4. Under Application Settings, click on Create.

If the web server were to go down unexpectedly, a reboot would be necessary. IIS is configured to start upon system start.

4.4 Summary

The maintenance section was created to help answer any questions that HazWeb's administer may have. A broad range of areas were covered ranging from the two part installation of ArcGIS Server and .NET ADF, configuring ArcGIS Server with the appropriate user account permissions, to backup logistics. Much of this chapter was described in methodical steps with illustrative screenshots of dialogs used throughout the installation and maintenance process. It is always the hope that nothing should happen to a system once it is up and running smoothly, but that cannot always be guaranteed. By having a detailed maintenance manual for the administrator, they are certain to have reliable information to fall back on when things do not work as planned.

CHAPTER FIVE

USER MANUAL

Any well-written software application should also include end user documentation. This documentation should explain the purpose of the application and details on how it works. Documentation should also be easily accessible, whether it is in hardcopy format or digital.

Since HazWeb is a web application accessible by anyone with an Internet connection, it would be ineffective to have hardcopy documentation describing how it works. To get around this limitation, a command was added to the HazWeb's start page that loads another web form. This form displays additional information describing HazWeb and how it should be used. This way, any person accessing this site, can easily get additional help by clicking on one simple command. The contents of the HazWeb's help page are listed in Appendix G.

CHAPTER SIX

CONCLUSION AND FUTURE DIRECTIONS

6.1 Conclusion

GIS is a technology that has been around for decades. It is now becoming more conventional with the onset of internet-based GIS applications. HazWeb is one example of how GIS can be used via the Internet. This project shows how any person wishing to determine hazard proximity can do so using standard GIS functionality via the Internet.

Real Estate Agents, insurance adjusters, existing and potential homebuyers and developers, are just a few sects of the public that could benefit from an application like this. Local governments can use an application like HazWeb to determine areas of imminent danger. Flood, fire, and earthquake faults are just a sampling of hazard data. For Southern Californians, it makes more sense to use these conditions. In places such as the South or Mid-West, other conditions may have higher precedence than something like earthquake fault lines.

6.2 Future Directions

Where will GIS and the Internet go in future directions? Undoubtedly, the outlook is good. Public and private sectors need the functionality provided in a GIS, in addition to the widespread audience attained through the Internet. Distributed GIS systems, such as HazWeb, are a useful tool for everyone. In the context of a homeowner, it provides a useful tool to display what may not already be known. For developers, it helps design what may or may not be an ideal place of development.

Future research considerations may be conducted for the HazWeb project in gathering and displaying detailed data concerning wind and flood areas. Topographical and wind hazard data and hazard prediction by using mathematical models based on data in a GIS system can be used to display regions that are affected by high winds. Similarly, mathematical modeling, soil data, and topographical data, along with a powerful computer engine, may be used to predict water and mud flows along areas near a hill or mountain side. Both these future research areas require extensive research, mathematics and experimentation.

Southern California has experienced many natural disasters in the past, and without question, will experience more as the years go on. Fires, floods, and earthquakes cannot be stopped. But with proper planning and informative decision-making tools, such as HazWeb, preparing for these types of disasters need not be as challenging as once thought.

APPENDIX A

ADDGRAPHICELEMENT CLASS PRINTOUT

```
using System;
using ESRI.ArcGIS.Server.WebControls;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Server;
using ESRI.ArcGIS.esriSystem;

namespace HazWeb
{
  /// <summary>
  /// Summary description for AddGraphicElement.
  /// </summary>
  internal class AddGraphicElement
  {
    public void AddElement(WebMap webMap, string[] StringPoint, string
          SpatialReference, string Text)
    {
      IMapServer mapServer = webMap.MapServer;
      IServerContext serverContext = webMap.ServerContext;
      IMapDescription mapDescription = webMap.MapDescription;
      if (webMap.IsPooled)
        webMap.ManageLifetime(serverContext);
      webMap.ManageLifetime(mapDescription);

      // Copying the geocode address point to the map CONTEXT
      IPoint point = serverContext.CreateObject("esriGeometry.Point") as IPoint;
      IGeometry geometry = point;
      webMap.ManageLifetime(point);
      webMap.ManageLifetime(geometry);
      geometry.SpatialReference = serverContext.LoadObject(SpatialReference) as ISpatialReference;
      point.PutCoords(Convert.ToDouble(StringPoint[0]),
                      Convert.ToDouble(StringPoint[1]));
      point.Project(mapDescription.SpatialReference);

      if (point == null)
        return;

      // Instantiate a graphics elements collection
      IGraphicElements graphicElements =
                      serverContext.CreateObject("esriCarto.GraphicElements") as
                      IGraphicElements;
      webMap.ManageLifetime(graphicElements);

      #region Address text. Create a TextSymbol and a TextElement, which will draw
                      the address string in the map
      // Specify symbol RGB color
      IRgbColor rgbColor = serverContext.CreateObject("esriDisplay.RgbColor") as
                      IRgbColor;
      webMap.ManageLifetime(rgbColor);

      if (Text != null)
      {
        stdole.IFontDisp font = serverContext.CreateObject("StdFont") as
                      stdole.IFontDisp;
        webMap.ManageLifetime(font);
        font.Name = "Arial";
        font.Size = 20;
        font.Bold = true;

        // Text symbol use to draw the Text element
        IFormattedTextSymbol formatedTextSymbol =
                      serverContext.CreateObject("esriDisplay.TextSymbol") as
                      IFormattedTextSymbol;
        webMap.ManageLifetime(formatedTextSymbol);
        formatedTextSymbol.Font = font;
        rgbColor.Red = 0;
        rgbColor.Green = 0;
        rgbColor.Blue = 0;
        formatedTextSymbol.Color = rgbColor;
        formatedTextSymbol.HorizontalAlignment =
                      esriTextHorizontalAlignment.esriTHALeft;
        formatedTextSymbol.VerticalAlignment =
                      esriTextVerticalAlignment.esriTVABottom;
        formatedTextSymbol.ShadowXOffset = 1;
        formatedTextSymbol.ShadowYOffset = -1;
```

```
    rgbColor.Red = 255;
    rgbColor.Green = 255;
    rgbColor.Blue = 255;
    formatedTextSymbol.ShadowColor = rgbColor;
    formatedTextSymbol.Position = esriTextPosition.esriTPSuperscript;

    // Text element to be added to the element in the map.
    ITextElement textElement =
                    serverContext.CreateObject("esriCarto.TextElement") as ITextElement;
    webMap.ManageLifetime(textElement);
    textElement.Text = Text;
    textElement.Symbol = formatedTextSymbol;

    // Specify the geometry or shape of the element to be drawn as a marker
        //          element
    IElement elementText = textElement as IElement;
    webMap.ManageLifetime(elementText);
    elementText.Geometry = point;
    IGraphicElement textGraphicElement = elementText as IGraphicElement;
    webMap.ManageLifetime(textGraphicElement);

    // Add the text element
    graphicElements.Add(textGraphicElement);
}
#endregion

#region Draw the select address candidate using a Red PUSH Pin Marker Element
// Specify type of marker symbol to represent the address point
// Add the Circle location graphic element
rgbColor.Red = 255;
rgbColor.Green = 255;
rgbColor.Blue = 255;
ISimpleMarkerSymbol simpleMarkerSymbol =
                    serverContext.CreateObject("esriDisplay.SimpleMarkerSymbol") as
                    ISimpleMarkerSymbol;
webMap.ManageLifetime(simpleMarkerSymbol);
simpleMarkerSymbol.Color = rgbColor;
simpleMarkerSymbol.Style = esriSimpleMarkerStyle.esriSMSCircle;
simpleMarkerSymbol.Size = 12;

IMarkerElement markerElement =
                    serverContext.CreateObject("esriCarto.MarkerElement") as IMarkerElement;
webMap.ManageLifetime(markerElement);
markerElement.Symbol = simpleMarkerSymbol;

IElement pointElement = markerElement as IElement;
webMap.ManageLifetime(pointElement);
pointElement.Geometry = point;
IGraphicElement markerGraphicElement = pointElement as IGraphicElement;
webMap.ManageLifetime(markerGraphicElement);

graphicElements.Add(markerGraphicElement);

// Add the Circle location graphic element
rgbColor.Red = 0;
rgbColor.Green = 0;
rgbColor.Blue = 0;
ISimpleMarkerSymbol simpleMarkerSymbol2 =
                    serverContext.CreateObject("esriDisplay.SimpleMarkerSymbol") as
                    ISimpleMarkerSymbol;
webMap.ManageLifetime(simpleMarkerSymbol2);
simpleMarkerSymbol2.Color = rgbColor;
simpleMarkerSymbol2.Style = esriSimpleMarkerStyle.esriSMSCircle;
simpleMarkerSymbol2.Size = 9;

IMarkerElement markerElement2 =
                    serverContext.CreateObject("esriCarto.MarkerElement") as IMarkerElement;
webMap.ManageLifetime(markerElement2);
markerElement2.Symbol = simpleMarkerSymbol2;

IElement pointElement2 = markerElement2 as IElement;
webMap.ManageLifetime(pointElement2);
pointElement2.Geometry = point;
IGraphicElement markerGraphicElement2 = pointElement2 as IGraphicElement;
webMap.ManageLifetime(markerGraphicElement2);
graphicElements.Add(markerGraphicElement2);
```

```
        #endregion

        // Pass the array of elements to the Graphic Container of the Map Description
        mapDescription.CustomGraphics = graphicElements;

        // Release server context
        serverContext.ReleaseContext();
    }
  }
}
```

APPENDIX B

BUFFER CLASS PRINTOUT

```
using System;
using System.Collections;
using System.Reflection;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Server.WebControls;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Server;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.Geodatabase;

namespace HazWeb
{
  /// <summary>
  /// Summary description for Buffer.
  /// </summary>
  public class Buffer
  {
    public Buffer() {}

    public IElement ChangeSymbol(WebMap webMap)
    {
      IServerContext serverContext = webMap.ServerContext;
      webMap.ManageLifetime(serverContext);

      // Return value
      IElement outElement = null;
      webMap.ManageLifetime(outElement);

      // Create a new color
      ESRI.ArcGIS.Display.IRgbColor newColor =
                    serverContext.CreateObject("esriDisplay.RGBColor") as
                    ESRI.ArcGIS.Display.IRgbColor;
      webMap.ManageLifetime(newColor);
      newColor.Red = 255;

      // Create new line symbol
      ILineSymbol lineSymbol =
                    serverContext.CreateObject("esriDisplay.SimpleLineSymbol") as
                    ILineSymbol;
      webMap.ManageLifetime(lineSymbol);
      lineSymbol.Color = newColor as ESRI.ArcGIS.Display.IColor;
      lineSymbol.Width = 1;

      // Create the new line fill symbol
      ILineFillSymbol lineFillSymbol =
                    serverContext.CreateObject("esriDisplay.LineFillSymbol") as
                    ILineFillSymbol;
      webMap.ManageLifetime(lineFillSymbol);
      lineFillSymbol.Color = newColor as ESRI.ArcGIS.Display.IColor;
      lineFillSymbol.Angle = 45;
      lineFillSymbol.LineSymbol = lineSymbol;
      lineFillSymbol.Outline = lineSymbol;
      lineFillSymbol.Separation = 5;

      // Get the graphics container
      IMap map = webMap.Map;
      webMap.ManageLifetime(map);
      IGraphicsContainer gc = map as IGraphicsContainer;
      gc.Reset();
      webMap.ManageLifetime(gc);

      // Enumerate through the graphic container
      IElement element = null;
      webMap.ManageLifetime(element);
      while ((element = gc.Next()) != null)
      {
        // Check for polygon element
        if (element.Geometry.GeometryType == esriGeometryType.esriGeometryPolygon)
        {
          // Check if element is type of fillshapeelement
          if (element is IFillShapeElement)
          {
            IFillShapeElement fillShapeElement = element as IFillShapeElement;
            webMap.ManageLifetime(fillShapeElement);
            fillShapeElement.Symbol = lineFillSymbol as IFillSymbol;
```

```
        // Set return value
        outElement = element;
      }
    }
  }
  // Release server context
  serverContext.ReleaseContext();

  // return value
  return outElement;
}

public IGeometry ChangeElementSymbol(WebMap webMap, out IElement polygonElement)
{
  IServerContext serverContext = webMap.ServerContext;
  webMap.ManageLifetime(serverContext);

  // Return value
  IGeometry outGeom = null;
  webMap.ManageLifetime(outGeom);

  // Create a new color
  ESRI.ArcGIS.Display.IRgbColor newColor =
                    serverContext.CreateObject("esriDisplay.RGBColor") as
                    ESRI.ArcGIS.Display.IRgbColor;
  webMap.ManageLifetime(newColor);
  newColor.Red = 255;

  // Create new line symbol
  ILineSymbol lineSymbol =
                    serverContext.CreateObject("esriDisplay.SimpleLineSymbol") as
                    ILineSymbol;
  webMap.ManageLifetime(lineSymbol);
  lineSymbol.Color = newColor as ESRI.ArcGIS.Display.IColor;
  lineSymbol.Width = 1;

  // Create the new line fill symbol
  ILineFillSymbol lineFillSymbol =
            serverContext.CreateObject("esriDisplay.LineFillSymbol") as
            ILineFillSymbol;
  webMap.ManageLifetime(lineFillSymbol);
  lineFillSymbol.Color = newColor as ESRI.ArcGIS.Display.IColor;
  lineFillSymbol.Angle = 45;
  lineFillSymbol.LineSymbol = lineSymbol;
  lineFillSymbol.Outline = lineSymbol;
  lineFillSymbol.Separation = 5;

  // get the map description
  IMapDescription mapDesc = webMap.MapDescription;
  webMap.ManageLifetime(mapDesc);

  // get the custom graphics
  IGraphicElements gE = mapDesc.CustomGraphics;
  webMap.ManageLifetime(gE);

  IElement outElement = null;
  webMap.ManageLifetime(outElement);

  int count = gE.Count;
  for (int i = 0; i < count; i++)
  {
    IElement element = gE.get_Element(i) as IElement;
    // Check if element is null
    if (element != null)
    {
      // Check if polygon geometry
      if (element.Geometry.GeometryType ==
                              esriGeometryType.esriGeometryPolygon)
      {
        // check if type of fill shape element
        if (element is IFillShapeElement)
        {
          IFillShapeElement fillShapeElement = element as IFillShapeElement;
          webMap.ManageLifetime(fillShapeElement);
          fillShapeElement.Symbol = lineFillSymbol as IFillSymbol;
```

```
                // Set out polygon element
                outElement = element;

                // Set return value
                outGeom = element.Geometry;
              } // fill shape element check
            } // geom check
          } // element != null
        } // for loop

        // Release server context
        serverContext.ReleaseContext();

        // out parameter
        polygonElement = outElement;

        // return value
        return outGeom;
      }

      public IGraphicElements CreateBuffer(WebMap webMap, double Distance)
      {
        try
        {
          IMapServer mapServer = webMap.MapServer;
          IServerContext serverContext = webMap.ServerContext;
          IMapDescription mapDescription = webMap.MapDescription;
          webMap.ManageLifetime(mapDescription);
          webMap.ManageLifetime(serverContext);

          // Before proceeding on, have to check if the custom graphics is null
          // if so, then exit
          if (mapDescription.CustomGraphics == null)
          {
            return null;
          }

          // Get the map description name
          string mapDescriptionName = mapDescription.Name;

          #region IFeatureCursorBuffer2 section
          IFeatureCursorBuffer2 featureCursorBuffer2 = serverContext.CreateObject("esriCarto.FeatureCursorBuffer") as
IFeatureCursorBuffer2;
          webMap.ManageLifetime(featureCursorBuffer2);
          // We don't have to pass in a featurecursor
          //featureCursorBuffer2.FeatureCursor = null;
          // Set the spatial reference
          //featureCursorBuffer2.SpatialReference = webMap.SpatialReference;
          featureCursorBuffer2.SpatialReference = mapDescription.SpatialReference;
          featureCursorBuffer2.BufferSpatialReference =
                        mapDescription.SpatialReference;
          // Pass in the data frame spatial reference
          featureCursorBuffer2.DataFrameSpatialReference =
                        mapDescription.SpatialReference;
          featureCursorBuffer2.SourceSpatialReference =
                        mapDescription.SpatialReference;
          featureCursorBuffer2.TargetSpatialReference =
                        mapDescription.SpatialReference;
          // Buffer's Distance
          featureCursorBuffer2.ValueDistance = Distance;
          // Map units and buffer units
          IMap map = webMap.Map;
          webMap.ManageLifetime(map);
          //esriUnits bufferUnits = (esriUnits) Enum.Parse(typeof(esriUnits), "5");
                        // 5 equates to Miles
          //featureCursorBuffer2.set_Units(map.MapUnits, bufferUnits);
          featureCursorBuffer2.set_Units(map.MapUnits, esriUnits.esriMiles);
          #endregion

          #region Parameters concerning the buffer
          IBufferProcessingParameter bufferProcessingParameter = featureCursorBuffer2
                        as IBufferProcessingParameter;
          webMap.ManageLifetime(bufferProcessingParameter);
          bufferProcessingParameter.AdjustCirclesForProjection = true;
          bufferProcessingParameter.SimplifyShapes = false;
```

```
bufferProcessingParameter.InputHasPolygons = false;
bufferProcessingParameter.BufferSpatialReference =
                esriBufferSpatialReferenceType.esriFeatureSetOptimizedSpatialReference;
bufferProcessingParameter.TargetSpatialReference =
                esriBufferSpatialReferenceType.esriMapSpatialReference;
bufferProcessingParameter.SaveAsGraphics = true;
#endregion

// Get the composite graphics layer
ICompositeGraphicsLayer compositeGraphicsLayer = map.ActiveGraphicsLayer as
                ICompositeGraphicsLayer;
webMap.ManageLifetime(compositeGraphicsLayer);

// MapServer requires Enumerator to be reset
IGraphicsContainer graphicsContainter = compositeGraphicsLayer as
                IGraphicsContainer;
webMap.ManageLifetime(graphicsContainter);
graphicsContainter.Reset();

// Get a reference to the custom graphics from
// the map description
IGraphicElements gE = mapDescription.CustomGraphics;

// Remove all polygon graphics
Functions.RemoveAllPolygonElementsFromGraphicsContainer(webMap);

// Enumerate through the point elements and have
// the first point element selected
for (int i = 0; i < gE.Count; i++)
{
  IElement ele = gE.get_Element(i) as IElement;
  webMap.ManageLifetime(ele);
  // Check if the element is a point. Not needed but since
  // we know that point elements are created but it's a
  // good check to have.
  if (ele.Geometry.GeometryType == esriGeometryType.esriGeometryPoint)
  {
    // Select the element
    IGraphicsContainerSelect gCS = graphicsContainter as
                        IGraphicsContainerSelect;
    webMap.ManageLifetime(gCS);
    gCS.SelectElement(ele);
    break;
  }
}

// Buffer to graphics
int selectedElements = 0;
bool hasPolygons = false;
featureCursorBuffer2.GraphicsLayer2(compositeGraphicsLayer, true, out
                selectedElements, out hasPolygons);
featureCursorBuffer2.BufferToGraphics(compositeGraphicsLayer);

// Create an instance of GraphicElement to store buffer graphics
IGraphicElements graphicBufferElements =
                serverContext.CreateObject("esriCarto.GraphicElements") as
                IGraphicElements;

// Get  the GraphicsContainerSelect to select all the graphic elements, loop through those, retrieved each geometry andd
// add it to the GeometryBag, which is going to be use to search features from the "Find" layer
IGraphicsContainerSelect graphicsContainerSelect = graphicsContainter as
                IGraphicsContainerSelect;
webMap.ManageLifetime(graphicsContainerSelect);
graphicsContainerSelect.SelectAllElements();
IEnumElement enumElement = graphicsContainerSelect.SelectedElements;
IElement element = null;
IGeometry geometry = null;
webMap.ManageLifetime(enumElement);
webMap.ManageLifetime(element);
webMap.ManageLifetime(geometry);

while ((element = enumElement.Next()) != null)
{
  geometry = element.Geometry;
  geometry.Project(mapDescription.SpatialReference);
  geometry.Project(mapDescription.SpatialReference);
```

```
            if (geometry.GeometryType == esriGeometryType.esriGeometryPolygon)
            {
              IGraphicElement graphicElement = element as IGraphicElement;
              webMap.ManageLifetime(graphicElement);
              graphicBufferElements.Add(graphicElement);
            }
          }
          // Release server context
          serverContext.ReleaseContext();

          return graphicBufferElements;
        }
        catch(System.Runtime.InteropServices.COMException e)
        {
          Functions.DisplayMessage(e.Message);
          return null;
        }
      }
    }
  }
}
```

APPENDIX C

DEFAULT CLASS PRINTOUT

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using Microsoft.VisualBasic;

namespace HazWeb
{
  /// <summary>
  /// Summary description for _Default.
  /// </summary>
  public class Default : System.Web.UI.Page
  {
    protected System.Web.UI.HtmlControls.HtmlTableCell VBTab;
    protected System.Web.UI.HtmlControls.HtmlTableCell VBNetTab;
    protected System.Web.UI.HtmlControls.HtmlInputText txtStreet;
    protected System.Web.UI.HtmlControls.HtmlInputText txtZIP;
    protected System.Web.UI.HtmlControls.HtmlInputText txtXCoord;
    protected System.Web.UI.HtmlControls.HtmlInputText txtYCoord;
    protected System.Web.UI.HtmlControls.HtmlInputButton btnLocate;
    protected System.Web.UI.HtmlControls.HtmlInputHidden txtHidden;

    private void Page_Load(object sender, System.EventArgs e)
    {
      if (!IsPostBack)
      {
        if (Session.IsNewSession)
        {
          // Clear textboxes
          ClearTextboxes();

          // Reset session variables
          ReleaseSessionVariables();

          // Set initial value of Value text to Address
          txtHidden.Value = "Address";
        }
      }
    }

    #region Web Form Designer generated code
    override protected void OnInit(EventArgs e)
    {
      //
      // CODEGEN: This call is required by the ASP.NET Web Form Designer.
      //
      InitializeComponent();
      base.OnInit(e);
    }

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      this.btnLocate.ServerClick += new
                        System.EventHandler(this.btnLocate_ServerClick);
      this.Load += new System.EventHandler(this.Page_Load);

    }
    #endregion

    private void btnLocate_ServerClick(object sender, System.EventArgs e)
    {
      #region Initial Checks, Saving to Session, and Redirect to Display.aspx
      // Check if Address tab has been clicked
      if (txtHidden.Value.ToUpper() == "ADDRESS")
      {
        // Before saving any data to the session,
```

```
    // check if the user entered valid
    // information

    // Check if zip code is valid with 5 digits
    if (Information.IsNumeric(txtZIP.Value.Trim().ToString()))
    {
      if (txtZIP.Value.Trim().Length != 5)
      {
        Functions.DisplayMessage("Please enter a 5 digit ZIP code");
        return;
      }
    }

    // Save to session
    Session["Street"] = txtStreet.Value.ToString();
    Session["City_ZIP"] = txtZIP.Value.ToString();
    Session["XCoord"] = "";
    Session["YCoord"] = "";
    Session["DataType"] = "Address";
  }

  // Check if Coordinates tab has bee clicked
  if (txtHidden.Value.ToUpper() == "COORDINATES")
  {
    // Before saving any data to the session,
    // check if the user entered valid
    // information

    // Check if the coordinates are numeric values
    if ((!Information.IsNumeric(txtXCoord.Value.Trim())) ||
                (!Information.IsNumeric(txtYCoord.Value.Trim())))
    {
      Functions.DisplayMessage("Please enter a numeric coordinate value");
      return;
    }

    // Save to session
    Session["Street"] = "";
    Session["City_Zip"] = "";
    Session["XCoord"] = txtXCoord.Value.ToString();
    Session["YCoord"] = txtYCoord.Value.ToString();
    Session["DataType"] = "Coordinates";
  }

  // Get then next page
  Page.Response.Redirect("Display.aspx");
  #endregion
}

#region Release Session Variables
// Release all session variables
private void ReleaseSessionVariables()
{
  Session["Buffers"] = null;
  Session["ZoomScale"] = null;
  Session["FullExtentScale"] = null;
  Session["ZoomLevel"] = null;
  Session["CurrentScale"] = null;
  Session["XCoord"] = null;
  Session["YCoord"] = null;
  Session["Street"] = null;
  Session["City_ZIP"] = null;
  Session["DataType"] = null;
  Session["ShapeField"] = null;
  Session["shapeArray"] = null;
  Session["SpatialReference"] = null;
  Session["stringPoint"] = null;
  Session["address"] = null;
  Session["GeocDS"] = null;
  Session["ZoomScale"] = null;
  Session["FullExtentScale"] = null;
  Session["ZoomLevel"] = null;
  Session["CurrentScale"] = null;
  Session["ErrorMessage"] = null;
  Session["Error"] = null;
}
```

```
    #endregion

    #region Clear Text box controls function
    private void ClearTextboxes()
    {
      // Clear out the text boxes
      txtStreet.Value = "";
      txtZIP.Value = "";
      txtXCoord.Value = "";
      txtYCoord.Value = "";
      txtHidden.Value = "";
    }
    #endregion
  }
}
```

APPENDIX D

DISPLAY CLASS PRINTOUT

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Xml;
using ESRI.ArcGIS.Server.WebControls;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Display;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Location;
using ESRI.ArcGIS.Server;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.DataSourcesGDB;
using Microsoft.VisualBasic;

namespace HazWeb
{
  /// <summary>
  /// Display class
  /// @inherits from System.Web.UI.Page
  /// </summary>
  public class Display : System.Web.UI.Page
  {
    #region Protected Class Members

    protected ESRI.ArcGIS.Server.WebControls.Toc HazWebTOC;
    protected ESRI.ArcGIS.Server.WebControls.GeocodeConnection JustZIPCodes;
    protected ESRI.ArcGIS.Server.WebControls.GeocodeConnection JustCities;
    protected ESRI.ArcGIS.Server.WebControls.GeocodeConnection StreetsZIPCodes;
    protected ESRI.ArcGIS.Server.WebControls.GeocodeConnection StreetsCities;
    protected System.Web.UI.WebControls.Label lblDataEntered;
    protected ESRI.ArcGIS.Server.WebControls.Map HazWebMap;
    protected System.Web.UI.WebControls.DataGrid HazWebDataGrid;
    protected System.Web.UI.WebControls.Label lblResultsCount;
    protected System.Web.UI.WebControls.Label lblSelectedItem;
    protected System.Web.UI.HtmlControls.HtmlInputText txtDistance;
    protected System.Web.UI.WebControls.Button btnFind;
    protected ESRI.ArcGIS.Server.WebControls.Impersonation HazWebImpersonation;
    protected System.Web.UI.WebControls.CheckBox chkBuffer;
    protected System.Web.UI.WebControls.LinkButton lnkHome;
    protected System.Web.UI.WebControls.Label Enter ;
    protected System.Web.UI.WebControls.Label lblNumberFound;
    protected System.Web.UI.WebControls.Table tableResults;
    protected System.Web.UI.WebControls.Label lblInfo;
    protected ESRI.ArcGIS.Server.WebControls.Toolbar HazWebToolBar;
    protected System.Web.UI.WebControls.DataGrid HazWebResultsGrid;
    protected System.Web.UI.HtmlControls.HtmlSelect ddlLayers;
    #endregion

    #region Private Class Members
    private System.ComponentModel.IContainer components;
    private string DATAGRID_OBJECTID = "FeatureID";
    private string DATAGRID_DISPLAYFIELD = "DisplayField";
    #endregion

    #region Private Enumeration for which geocode object to use
    private enum GeoCodeType : long
    {
      NONE = -1,
      JUST_ZIP_CODE = 0,
      JUST_CITIES= 1,
      STREETS_ZIP_CODES = 2,
      STREETS_CITY_NAME = 3,
      COORDINATES = 4
    }
    #endregion

    #region Web Form Designer generated code
    override protected void OnInit(EventArgs e)
```

```
{
  //
  // CODEGEN: This call is required by the ASP.NET Web Form Designer.
  //
  InitializeComponent();
  base.OnInit(e);
}

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
  this.components = new System.ComponentModel.Container();
  this.JustZIPCodes = new
                ESRI.ArcGIS.Server.WebControls.GeocodeConnection(this.components);
  this.JustCities = new
                ESRI.ArcGIS.Server.WebControls.GeocodeConnection(this.components);
  this.StreetsZIPCodes = new
                ESRI.ArcGIS.Server.WebControls.GeocodeConnection(this.components);
  this.StreetsCities = new
                ESRI.ArcGIS.Server.WebControls.GeocodeConnection(this.components);
  //
  // JustZIPCodes
  //
  this.JustZIPCodes.Host = "sandbox";
  this.JustZIPCodes.ServerObject = "HazWebZipCodes";
  //
  // JustCities
  //
  this.JustCities.Host = "sandbox";
  this.JustCities.ServerObject = "HazWebCityGeocode";
  //
  // StreetsZIPCodes
  //
  this.StreetsZIPCodes.Host = "sandbox";
  this.StreetsZIPCodes.ServerObject = "HazWebGeocode";
  //
  // StreetsCities
  //
  this.StreetsCities.Host = "sandbox";
  this.StreetsCities.ServerObject = "HazWebStreetsCityGeocode";
  this.HazWebResultsGrid.ItemCommand += new
        System.Web.UI.WebControls.DataGridCommandEventHandler(this.HazWebResultsGrid_ItemCommand);
  this.HazWebResultsGrid.PageIndexChanged += new
        System.Web.UI.WebControls.DataGridPageChangedEventHandler(this.HazWebResultsGrid_PageIndexChanged);
  this.HazWebDataGrid.ItemCommand += new
        System.Web.UI.WebControls.DataGridCommandEventHandler(this.HazWebDataGrid_ItemCommand);
  this.HazWebDataGrid.PageIndexChanged += new
        System.Web.UI.WebControls.DataGridPageChangedEventHandler(this.HazWebDataGrid_PageIndexChanged);
  this.HazWebToolBar.CommandClick += new
        ESRI.ArcGIS.Server.WebControls.ToolbarCommandClickEventHandler(this.HazWebToolBar_CommandClick);
  this.btnFind.Click += new System.EventHandler(this.btnFind_Click);
  this.lnkHome.Click += new System.EventHandler(this.lnkHome_Click);
  this.chkBuffer.CheckedChanged += new System.EventHandler(this.chkBuffer_CheckedChanged);
  this.Load += new System.EventHandler(this.Page_Load);

}
#endregion

#region Private functions

#region Geocoding
// Its required to persist the DataGrid columns in each post back.
// Property that will indicate if the columns have been added to the grid and its get persisted in the view state
private bool DynamicColumnAdded
{
  get
  {
    object bAdded = ViewState["DynamicColumnAdded"];
    return (bAdded == null) ? false : true;
  }

  set
  {
```

```
    ViewState["DynamicColumnAdded"] = value;
  }
}

private bool DynamicResultsColumnAdded
{
  get
  {
    object bAdded = ViewState["DynamicResultsColumnAdded"];
    return (bAdded == null) ? false : true;
  }
  set
  {
    ViewState["DynamicResultsColumnAdded"] = value;
  }
}

// Restore DataGrid columns each time the page runs
protected override void LoadViewState(object savedState)
{
  base.LoadViewState(savedState);

  if (DynamicColumnAdded)
  {
    this.AddColumns();
  }

  if (DynamicResultsColumnAdded)
  {
    this.AddResultsColumns();
  }
}

protected override object SaveViewState()
{
  // Save State as an acumulative array of objects.
  object baseState = base.SaveViewState();

  // If the Map control is visible, make HTML controls visible as well
  Response.Write(Functions.IsMapVisible(HazWebMap));

  if (tableResults.Visible)
  {
    Functions.CreateTableResults((DataTable)Session["FeatureResults"], ref tableResults);
  }

  return baseState;
}

private void GeocodeAddress(int index)
{
  if (Session["shapeArray"] == null)
  {
    return;
  }

  ArrayList shapeArray = new ArrayList((ArrayList) Session["shapeArray"]);
  string geocodedPoint = shapeArray[index].ToString();

  if (geocodedPoint.Length == 0)
  {
    return;
  }

  char[] sep = {','};
  string[] stringPoint = geocodedPoint.Split(sep);

  try
  {
    using (WebMap webMap = HazWebMap.CreateWebMap())
    {
      IMapServer mapServer = webMap.MapServer;
      IServerContext serverContext = webMap.ServerContext;
      IMapDescription mapDescription = webMap.MapDescription;
      webMap.ManageLifetime(serverContext);
      webMap.ManageLifetime(mapDescription);
```

```
        // Copying the geocode address point to the map CONTEXT
        IPoint point = serverContext.CreateObject("esriGeometry.Point") as
                                    IPoint;
        IGeometry geometry = point as IGeometry;
        webMap.ManageLifetime(point);
        webMap.ManageLifetime(geometry);
        string spatialReferenceXML = Session["SpatialReference"] as String;
        geometry.SpatialReference =
                                    serverContext.LoadObject(spatialReferenceXML) as
                                    ISpatialReference;
        point.PutCoords(Convert.ToDouble(stringPoint[0]),
                                    Convert.ToDouble(stringPoint[1]));
        // need to project the point before center At
        point.Project(mapDescription.SpatialReference);

        if (point != null)
        {
          // Store point as string  in session
          Session["stringPoint"] = stringPoint;

          string x1 = String.Format("{0:N6}", point.X);
          string y1 = String.Format("{0:N6}", point.Y);

          lblSelectedItem.Text = "Coordinates -  x: " + x1 + ", y: " + y1;
          lblSelectedItem.Visible = true;

          // Add n geocode address point as a graphic element to the Map Description in the map CONTEXT
          AddGraphicElement graphicElement = new AddGraphicElement();
          graphicElement.AddElement(webMap, stringPoint, spatialReferenceXML,
                                    (string) Session["address"]);

          // Redraw the map
          #region Zoom to location
          IMapArea mapArea = mapDescription.MapArea;
          ICenterAndScale centerScale =
                                    serverContext.CreateObject("esriCarto.CenterAndScale") as
                                    ICenterAndScale;
          webMap.ManageLifetime(mapArea);
          webMap.ManageLifetime(centerScale);
          centerScale.Center = point;
          centerScale.MapScale = (double) Session["ZoomScale"];
          mapDescription.MapArea = centerScale as IMapArea;
          #endregion

          // Refresh the map
          webMap.Refresh();
          UpdateUI(true);
        }
        serverContext.ReleaseContext();
      }
    }
    catch (Exception error)
    {
      callErrorPage("Address could not be geocoded.", error);
    }
}

// Adding columns to the DataGrid dynamically
private void AddColumns()
{
  if (Session["GeocDS"] == null)
    return;

  // Use candidate fields retrieved from the Geocoding server object to create a bound column in the DataGird control
  DataSet candidateFieldsDataSet =  (DataSet) Session["GeocDS"];
  DataTable dataTable = candidateFieldsDataSet.Tables[0];

  string colName = null;
  for (int i = 0; i < dataTable.Columns.Count; i ++ )
  {
    colName = dataTable.Columns[i].ColumnName;

    if ((colName.ToUpper() == "SCORE") || (colName.ToUpper() == "STREETNAME")
                || (colName.ToUpper() == "LEFTZONE"))
    {
```

```
        BoundColumn bColumn = new BoundColumn();
        bColumn.DataField = colName;
        if (colName.ToUpper() == "LEFTZONE")
        {
          bColumn.HeaderText = "ZIP";
        }
        else
        {
          bColumn.HeaderText = colName;
        }
        HazWebDataGrid.Columns.Add(bColumn);
      }
    }

    // Adding an extra-column to allow the user to center the map in a specific feature
    ButtonColumn btnColumn = new ButtonColumn();
    btnColumn.ButtonType = ButtonColumnType.PushButton;
    btnColumn.Text = "Map";
    btnColumn.HeaderText = "Locate";
    btnColumn.CommandName = "FindCandidate";
    HazWebDataGrid.Columns.Add(btnColumn);

    this.DynamicColumnAdded = true;
  }
  #endregion

  #region Geocode UI
  // Toggle visibility of controls
  private void UpdateUI(bool bUpdate)
  {
    HazWebMap.Visible = bUpdate;
    HazWebDataGrid.Visible = bUpdate;
    HazWebToolBar.Visible = bUpdate;
    HazWebTOC.Visible = bUpdate;
    lblResultsCount.Visible = bUpdate;
    lblSelectedItem.Visible = bUpdate;
  }

  #region Add Columns in the HazWebResult Data grid
  public void AddResultsColumns()
  {
    // Adding columns  to the DataGrid dynamically
    // These are hardwire columns returned from the WebMap::Find
    // Feature ID
    BoundColumn bColumn = new BoundColumn();
    bColumn.DataField = DATAGRID_OBJECTID;
    bColumn.Visible = false;
    HazWebResultsGrid.Columns.Add(bColumn);

    // Adding an extra-column to allow the user to get detailed info from a specific feature
    ButtonColumn btnColumn = new ButtonColumn();
    btnColumn.HeaderText = "Features found";
    btnColumn.ButtonType = ButtonColumnType.LinkButton;
    btnColumn.ItemStyle.ForeColor = Color.RoyalBlue;
    btnColumn.DataTextField = DATAGRID_DISPLAYFIELD;
    btnColumn.CommandName = "IDFeature";
    btnColumn.ItemStyle.Wrap = false;
    HazWebResultsGrid.Columns.Add(btnColumn);

    // Adding an extra-column to allow the user to center the map in a specific feature
    btnColumn = new ButtonColumn();
    btnColumn.ButtonType = ButtonColumnType.PushButton;
    btnColumn.Text = "Map";
    btnColumn.HeaderText = "Locate Feature";
    btnColumn.CommandName = "FindFeature";
    btnColumn.ItemStyle.Wrap = false;
    HazWebResultsGrid.Columns.Add(btnColumn);

    this.DynamicResultsColumnAdded = true;
  }
  #endregion

  #endregion

  #region Release Session Variables
  // Release all session variables
```

```
private void ReleaseSessionVariables()
{
  Session["selectedDataSet"] = null;
  Session["Buffers"] = null;
  Session["ZoomScale"] = null;
  Session["FullExtentScale"] = null;
  Session["ZoomLevel"] = null;
  Session["CurrentScale"] = null;
  Session["XCoord"] = null;
  Session["YCoord"] = null;
  Session["Street"] = null;
  Session["City_ZIP"] = null;
  Session["DataType"] = null;
  Session["ShapeField"] = null;
  Session["shapeArray"] = null;
  Session["SpatialReference"] = null;
  Session["stringPoint"] = null;
  Session["address"] = null;
  Session["GeocDS"] = null;
  Session["ZoomScale"] = null;
  Session["FullExtentScale"] = null;
  Session["ZoomLevel"] = null;
  Session["CurrentScale"] = null;
  Session["ErrorMessage"] = null;
  Session["Error"] = null;
  Session["FidSet"] = null;
  Session["LayerID"] = null;
  Session["FeatureResults"] = null;
}
#endregion

#region Get the address from the geocode object
private void RetrieveGeoCodingRecords(long lGeoCodeType)
{
  if (lGeoCodeType == (long)GeoCodeType.STREETS_ZIP_CODES)
  {
    DisplayGeoCodeRecords(StreetsZIPCodes, lGeoCodeType);
  }
  else if (lGeoCodeType == (long)GeoCodeType.STREETS_CITY_NAME)
  {
    DisplayGeoCodeRecords(StreetsCities, lGeoCodeType);
  }
  else if (lGeoCodeType == (long)GeoCodeType.JUST_CITIES)
  {
    DisplayGeoCodeRecords(JustCities, lGeoCodeType);
  }
  else if (lGeoCodeType == (long)GeoCodeType.JUST_ZIP_CODE)
  {
    DisplayGeoCodeRecords(JustZIPCodes, lGeoCodeType);
  }
  else
  {
    // Should never get here
    DisplayGeoCodeRecords(null, lGeoCodeType);
  }
}

private void CandidatesXY(DataSet Candidates)
{
  // Store the geocoded shape x,y coords in an array
  ArrayList shapeArray = new ArrayList();

  // Get the candidates table
  DataTable dataTable = Candidates.Tables[0];

  // Find the shape field column index
  string shapeField = Session["ShapeField"] as String;
  int shapeIndex = dataTable.Columns[shapeField].Ordinal;

  // retreive the X,Y coord from XML
  XmlNodeList xmlNodeList = null;
  foreach (DataRow row in Candidates.Tables[0].Rows)
  {
    xmlNodeList = (row[shapeIndex] as XmlNode).ChildNodes;
    // Get the X and Y values from the Geometry XML
    shapeArray.Add(xmlNodeList[0].InnerText + ", " +
```

```
                    xmlNodeList[1].InnerText);
  }

  // Save the shapeArray to the session
  Session["shapeArray"] = shapeArray;
}

private void DisplayGeoCodeRecords(GeocodeConnection newGeoCodeConnection, long
        lgeoType)
{
  // Get the information entered by the user
  string sXCoord = Session["XCoord"].ToString();
  string sYCoord = Session["YCoord"].ToString();
  string sStreet = Session["Street"].ToString();
  string sZIP = Session["City_ZIP"].ToString();
  string sDataType = Session["DataType"].ToString();
  string sCurrentAddress = "";

  try
  {
    // Check if we're dealing with user input address or coordinates
    if ((sDataType.ToUpper() == "ADDRESS") && (newGeoCodeConnection != null))
    {
      using (WebGeocode geocodeHelper =
                                newGeoCodeConnection.CreateWebGeocode())
      {
        // Get the geocoding server context
        IServerContext geocodeServerContext = geocodeHelper.ServerContext;
        if (geocodeHelper.IsPooled)
        {
          geocodeHelper.ManageLifetime(geocodeServerContext);
        }

        if (geocodeServerContext == null)
        {
          return;
        }

        // Create a property set to hold the geocoding properties
        IPropertySet propertySet =
                                geocodeServerContext.CreateObject("esriSystem.PropertySet") as
                                IPropertySet;
        geocodeHelper.ManageLifetime(propertySet);

        // Set up the property set
        if ((lgeoType == (long)GeoCodeType.STREETS_CITY_NAME) || (lgeoType ==
                                (long)GeoCodeType.STREETS_ZIP_CODES))
        {
          propertySet.SetProperty("Street", sStreet);
          propertySet.SetProperty("Zone", sZIP);
          sCurrentAddress = sStreet + ", " + sZIP;
        }
        else if (lgeoType == (long)GeoCodeType.JUST_CITIES)
        {
          propertySet.SetProperty("City", sZIP);
          propertySet.SetProperty("State", "California");
          sCurrentAddress = sZIP;
        }
        else if (lgeoType == (long)GeoCodeType.JUST_ZIP_CODE)
        {
          propertySet.SetProperty("ZIP", sZIP);
          sCurrentAddress = sZIP;
        }

        // Retrieve a recordset with the address matched candidates
        DataSet candidatesDataSet =
                                geocodeHelper.FindAddressCandidates(propertySet, null, false,
                                true);

        #region Bound/unbound dataset to the DataGrid
        if (candidatesDataSet != null)
        {
          if (candidatesDataSet.Tables[0].Rows.Count > 0)
          {
            // Store the dataset in session, it's re-used when the user changes page in the data grid control
            Session["GeocDS"] = candidatesDataSet;
```

```
            if (!DynamicColumnAdded)
            {
              this.AddColumns();
            }
            HazWebDataGrid.DataSource = candidatesDataSet;
            HazWebDataGrid.DataBind();
            HazWebDataGrid.Visible = true;
            lblResultsCount.Text = "Records found: " +
                                        candidatesDataSet.Tables[0].Rows.Count.ToString();
            lblResultsCount.Visible = true;

            // Get a X, Y array of geocoded addresses from the returned candidates dataset
            CandidatesXY(candidatesDataSet);

            // Save the current address to a session variable
            Session["address"] = sCurrentAddress;
          }
          else
          {
            // no records found
            HazWebDataGrid.DataSource = "";
            UpdateUI(false);
            lblDataEntered.Text = "Address could not be match: " +
                                        sCurrentAddress;
          }
        }
        else
        {
          HazWebDataGrid.DataSource = "";
          UpdateUI(false);
          lblDataEntered.Text = "Address could not be matched: " +
                                sCurrentAddress;
        }
        #endregion
      }
    }
      // User entered coordinates
    else
    {
      ArrayList arrList = new ArrayList();
      arrList.Add(sXCoord + ", " + sYCoord);
      Session["shapeArray"] = arrList;
    }
  }
  catch
  {
    HazWebDataGrid.DataSource = "";
    UpdateUI(false);
    lblDataEntered.Text = "Address could not be matched: " + sCurrentAddress;
  }
}
#endregion

#region Find selected features information
private void FindFeatureDetails(int layerID, int featureID)
{
  try
  {
    // Table gets writen in SaveViewState to mantain it between postbacks
    DataTable resultsDataTable = Functions.SearchFeature(HazWebMap, layerID,
                featureID);
    Session["FeatureResults"] = resultsDataTable;

    if (!tableResults.Visible)
    {
      tableResults.Visible = true;
    }
  }
  catch (Exception error)
  {
    callErrorPage("Error finding feature attributes.", error);
  }
}

#endregion
```

```
#region Retrieve information on focus layer
private void GetZoomInfoOnLayers()
{
  try
  {
    // Retrieve layers from current Data Frame
    using ( WebMap webMap = HazWebMap.CreateWebMap() )
    {
      IMapServer mapServer = webMap.MapServer;
      IServerContext serverContext = webMap.ServerContext;
      webMap.ManageLifetime(serverContext);

      #region Finding ArcMap ZoomToFeature scale
      IMapServerInfo mapServerInfo =
                                  mapServer.GetServerInfo(webMap.DataFrame);
      IEnvelope fullExtent = mapServerInfo.FullExtent;
      webMap.ManageLifetime(mapServerInfo);
      webMap.ManageLifetime(fullExtent);

      IMapDescription mapDescription = webMap.MapDescription;
      IMapExtent mapExtent = mapDescription.MapArea as IMapExtent;
      webMap.ManageLifetime(mapDescription);
      webMap.ManageLifetime(mapExtent);
      mapExtent.Extent = fullExtent;
      mapDescription.MapArea = mapExtent as IMapArea;

      IImageDisplay imageDisplay =
                                  webMap.ImageDescriptor.QueryImageDisplay(serverContext);
      webMap.ManageLifetime(imageDisplay);

      // Finding an arbitrary scale to ZoomToFeature
      double scale = (mapServer.ComputeScale(mapDescription, imageDisplay)) /
                                  20;
      Session["Scale"] = scale;
      #endregion

      // Arraylist to store each layer best zoom scale
      ArrayList layersVisibleScale = new ArrayList();

      // Loop through MapLayersInfos to retrieve MapLayerInfo::Name and
                                  MapLayerInfo::ID
      // also, to find best scale to ZoomToFeature
      IMapLayerInfos mapLayerInfos = mapServerInfo.MapLayerInfos;
      webMap.ManageLifetime(mapLayerInfos);
      for (int i = 0; i < mapLayerInfos.Count; i ++)
      {
        if (mapLayerInfos.get_Element(i).Type == "Feature Layer")
        {
          #region Build an arraylist of best scale to zoom to feature
                                  according to the layer's visible scale range
          double minScale = mapLayerInfos.get_Element(i).MinScale;
          double maxScale = mapLayerInfos.get_Element(i).MaxScale;

          // Range of scales at which this layer will be shown
          // minScale = Out beyond  and  maxScale = In beyond
          // minScale should be always greater than maxScale
          if (minScale > 0 && maxScale == 0)
          {
            // if scale is used, layer will not be visible,
            // therefore set the scale to be minScale
            // in order to see features in layer when ZoomToFeature
            if (scale > minScale)
              scale = minScale;
          }
          else if (minScale == 0 && maxScale > 0)
          {
            // if scale is used, layer will not be visible,
            // therefore set the scale to be maxScale
            // in order to see features in layer when ZoomToFeature
            if (scale < maxScale)
              scale = maxScale;
          }
          else if (minScale > 0 && maxScale > 0)
          {
            // if layer has min scale and max scale,
            // use an average
```

```
            if (!(scale < minScale && scale > maxScale))
              scale = maxScale + (minScale - maxScale) / 2;
          }

          // Keep layer's best visible scale in an ArrayList
          layersVisibleScale.Add(scale);
          #endregion
        }
      }

      // Store arraylist of each layer best zoom scale
      Session["layersVisibleScale"] = layersVisibleScale;
    }
  }
  catch (Exception error)
  {
    callErrorPage("Application can not retrieve map layers.", error);
  }
}

#endregion

#region Zoom to specific feature
private void ZoomToFeature(int layerID, int featureID)
{
  try
  {
    using (WebMap webMap = HazWebMap.CreateWebMap())
    {
      #region Find the BEST scale to zoom to
      double scale = (double) Session["Scale"];
      // Get Layer's scale
      int prevLayerID = -1;
      if (Session["LayerID"] != null)
      {
        prevLayerID = (int) Session["LayerID"];
      }

      if (prevLayerID != layerID)
      {
        ArrayList layersVisibleScale = new ArrayList((ArrayList)
                                    Session["layersVisibleScale"]);
        scale = (double) layersVisibleScale[prevLayerID];
      }
      #endregion

      IMapServer mapServer = webMap.MapServer;
      IServerContext serverContext = webMap.ServerContext;
      webMap.ManageLifetime(serverContext);

      // Create a new FIDSet and add the FeatureID to zoom to
      IFIDSet fidSet = serverContext.CreateObject("esriGeodatabase.FIDSet") as
                                    IFIDSet;
      webMap.ManageLifetime(fidSet);
      fidSet.Add(featureID);

      // Use Feature Extent to zoom to feature
      IFeatureExtent featureExtent =
                                    serverContext.CreateObject("esriCarto.FeatureExtent") as
                                    IFeatureExtent;
      webMap.ManageLifetime(featureExtent);
      // Specify DataFrame name
      featureExtent.MapName = webMap.MapDescription.Name;
      // Specify Layer
      featureExtent.LayerID = layerID;
      // Ratio to expand the feature extent envelope
      featureExtent.ExpandRatio = 1.25;
      // Specify scale
      featureExtent.DefaultScale = scale;
      // Pass FIDSet with wanted zoom to feature feature IDs
      featureExtent.FeatureIDs = fidSet;

      // Zoom to feature extent
      webMap.DrawExtent(featureExtent);

      // Store layer ID
```

```
        Session["LayerID"] = layerID;

        // remove server context
        serverContext.ReleaseContext();
      }
    }
    catch (Exception error)
    {
      callErrorPage("Error zooming to feature.", error);
    }
  }

  #endregion

  #region ERROR handling
  /// <summary>
  /// Displays the error page.
  /// </summary>
  /// <param name="errorMessage">The error message to display.</param>
  /// <param name="exception">The exception that was caught.</param>
  private void callErrorPage(string errorMessage, Exception exception)
  {
    Session["ErrorMessage"] = errorMessage;
    Session["Error"] = exception;
    Page.Response.Redirect("ErrorPage.aspx",true);
  }
  #endregion

  #endregion

  #region Private Web Control Events

  /// <summary>
  /// Handles unhandled exceptions in the page.
  /// </summary>
  private void Page_Error(object sender, System.EventArgs e)
  {
    Exception exception = Server.GetLastError();
    Server.ClearError();
    callErrorPage("Page_Error", exception);
  }

  private void Page_Load(object sender, System.EventArgs e)
  {
    #region Check parameters of MapControl
    // If there is no Host or ServerObject defined,
    // then there is no point to continuing since there will be no map to display
    if ((HazWebMap.Host == null) || (HazWebMap.Host == ""))
    {
      callErrorPage("Host property not defined for the Map control.", null);
    }

    if ((HazWebMap.ServerObject == null) || (HazWebMap.ServerObject == "") ||
                        (HazWebMap.ServerObject == "(none)"))
    {
      callErrorPage("ServerObject property not defined for the Map control",
                        null);
    }

    // check if the server object can be accessed
    ServerConnection connection = HazWebMap.ServerConnection;
    if (connection == null)
    {
      callErrorPage("Invalid server connection.", null);
    }
    #endregion

    if (!IsPostBack)
    {

      if (!Session.IsNewSession)
      {
        // Hide all controls
        UpdateUI(false);

        // get layers zoom area
```

```
GetZoomInfoOnLayers();

#region Getting full extent scale use to calculate scale interval for
                        fixed zoom
using (WebMap webMap = HazWebMap.CreateWebMap())
{
  // Get the map server
  IMapServer mapServer = webMap.MapServer;
  // Get the server context
  IServerContext serverContext = webMap.ServerContext;
  if (webMap.IsPooled)
  {
    webMap.ManageLifetime(serverContext);
  }

  // Get the map Server info
  IMapServerInfo mapServerInfo =
                        mapServer.GetServerInfo(webMap.DataFrame);
  webMap.ManageLifetime(mapServerInfo);

  // Get the map description
  IMapDescription mapDescription = webMap.MapDescription;
  webMap.ManageLifetime(mapDescription);

  // Get the full extent of the map
  IEnvelope fullExtent = mapServerInfo.FullExtent;
  IMapExtent mapExtent = mapDescription.MapArea as IMapExtent;
  webMap.ManageLifetime(fullExtent);
  webMap.ManageLifetime(mapExtent);
  mapExtent.Extent = fullExtent;
  mapDescription.MapArea = mapExtent as IMapArea;

  // Get the image display
  IImageDisplay imageDisplay =
                        webMap.ImageDescriptor.QueryImageDisplay(serverContext);
  webMap.ManageLifetime(imageDisplay);

  // Compute the scale
  double scale = mapServer.ComputeScale(mapDescription, imageDisplay);
  // Store scale factor to Zoom to feature
  Session["ZoomScale"] = scale / 100;
  // Store full extent scale in session
  Session["FullExtentScale"] = scale;
  // Store current fixed zoom level
  Session["ZoomLevel"] = -1;
  // Store current map scale
  Session["CurrentScale"] = null;
}
#endregion

#region Find geocoding address candidates fields
// Get candidate fields to build DataGrid
using (WebGeocode geocodeHelper = StreetsCities.CreateWebGeocode())
{
  // Get the geocoding server context
  IServerContext geocodeServerContext = geocodeHelper.ServerContext;
  geocodeHelper.ManageLifetime(geocodeServerContext);

  if (geocodeServerContext == null)
    return;

  IGeocodeServer geocodeServer = geocodeHelper.GeocodeServer;
  // Find SHAPE field
  IFields fields = geocodeServer.GetResultFields(null);
  IField field = null;
  geocodeHelper.ManageLifetime(fields);
  geocodeHelper.ManageLifetime(field);
  for (int i = 0; i < fields.FieldCount; i ++)
  {
    field = fields.get_Field(i);
    if (field.Type == esriFieldType.esriFieldTypeGeometry)
      break;
  }

  // Shape field
  Session["ShapeField"] = field.Name;
```

```
    // Get the LOCATOR's spatial reference
    string locatorSpatialReferenceXML = geocodeServerContext.SaveObject
                                        (field.GeometryDef.SpatialReference);
    // and keep it in session, it's need it when user wants to draw the location
    Session["SpatialReference"] = locatorSpatialReferenceXML;
}
#endregion

#region Initial Checks for User Data from Default.aspx
// Check if the XCoord, YCoord, ZIP, DataType and Street sessions are populated
string sXCoord = Session["XCoord"].ToString();
string sYCoord = Session["YCoord"].ToString();
string sStreet = Session["Street"].ToString();
string sZIP = Session["City_ZIP"].ToString();
string sDataType = Session["DataType"].ToString();

// If the streets and zip were populated, then we know that
// the user entered an address
if (sDataType.ToUpper() == "ADDRESS")
{
    // Check for a valid ZIP code
    bool IsValidZIP = Information.IsNumeric(sZIP);

    // Check if the zip is numeric
    // and street is populated
    if ((sStreet.Length > 0) && (IsValidZIP))
    {
        // Streets with ZIP codes
        lblDataEntered.Text = "Address: " + sStreet + ", " + sZIP;
        RetrieveGeoCodingRecords((long)GeoCodeType.STREETS_ZIP_CODES);
    }
        // Street is populated but ZIP is city name
    else if ((sStreet.Length > 0) && (!IsValidZIP))
    {
        // Streets with City name
        lblDataEntered.Text = "Address: " + sStreet + ", " + sZIP;
        RetrieveGeoCodingRecords((long)GeoCodeType.STREETS_CITY_NAME);
    }
        // No street info but ZIP is city name
    else if ((sStreet.Trim().Length == 0) && (!IsValidZIP))
    {
        // Just City name
        lblDataEntered.Text = "City Name: " + sZIP;
        RetrieveGeoCodingRecords((long)GeoCodeType.JUST_CITIES);
    }
        // No street info but valid ZIP
    else if ((sStreet.Trim().Length == 0) && (IsValidZIP))
    {
        // Just ZIP Code
        lblDataEntered.Text = "ZIP Code: " + sZIP;
        RetrieveGeoCodingRecords((long)GeoCodeType.JUST_ZIP_CODE);
    }
}

// User decided to enter coordinates
if (sDataType.ToUpper() == "COORDINATES")
{
    // Check if the x and y coordinates are populated
    if ((sXCoord.Length > 0) && (sYCoord.Length > 0))
    {
        // Coordinates
        lblDataEntered.Text = "Coordinates: " + sXCoord + ", " + sYCoord;
        RetrieveGeoCodingRecords((long)GeoCodeType.COORDINATES);
        // Save the address to the address session
        Session["address"] = sXCoord + ", " + sYCoord;
        // Geocode the first record - it will always be the first index
        GeocodeAddress(0);
        // update ui
        UpdateUI(true);
    }
}

// Geocode the location
if (HazWebDataGrid.Items.Count > 0)
{
    GeocodeAddress(0);
```

```
      }
      #endregion

    }
  }
}

private void HazWebToolBar_CommandClick(object sender,
        ESRI.ArcGIS.Server.WebControls.ToolbarCommandClickEventArgs args)
{
  try
  {
    switch(args.CommandName)
    {
      case "fe":      // Full Extent Command
        using (WebMap webMap = HazWebMap.CreateWebMap())
        {
          //webMap.DrawFullExtent();
          Functions.ZoomToRoadsExtent(webMap);
        }
        break;
      default:
        break;
    }
  }
  catch(Exception error)
  {
    callErrorPage(String.Format("Error in {0} command.", args.CommandName),
                  error);
  }
}

private void HazWebDataGrid_ItemCommand(object source,      System.Web.UI.WebControls.DataGridCommandEventArgs e)
{
  try
  {
    // if user clicks on DataGrid Map button, the candidate will be drawn in the map
    if (e.Item.ItemType == ListItemType.Item || e.Item.ItemType ==
                  ListItemType.AlternatingItem || e.Item.ItemType ==
                  ListItemType.SelectedItem)
    {
      DataGridItem item = e.Item;
      HazWebDataGrid.SelectedIndex = item.ItemIndex;

      if (e.CommandName == "FindCandidate")
      {
        // Uncheck the checkbox
        if (chkBuffer.Checked)
        {
          chkBuffer.Checked = false;
        }

        // Geocode what was selected
        GeocodeAddress(item.DataSetIndex);
      }
    }
  }
  catch (Exception error)
  {
    callErrorPage(String.Format("DataGrid error in {0} command.",
                  e.CommandName), error);
  }
}

private void HazWebDataGrid_PageIndexChanged(object source,
        System.Web.UI.WebControls.DataGridPageChangedEventArgs e)
{
  try
  {
    // USER clicks on a particular page index
    HazWebDataGrid.CurrentPageIndex = e.NewPageIndex;

    if (Session["GeocDS"] == null)
      return;

    System.Data.DataSet dataSet = Session["GeocDS"] as DataSet;
```

```
    HazWebDataGrid.DataSource = dataSet;
    HazWebDataGrid.DataBind();
  }
  catch (Exception error)
  {
    callErrorPage(String.Format("DataGrid error in {0} page index.",
                  e.NewPageIndex), error);
  }
}

private void lnkHome_Click(object sender, System.EventArgs e)
{
  // Remove session variables
  ReleaseSessionVariables();
  // Redirect the page back to Default.aspx
  Page.Response.Redirect("Default.aspx");
}

private void HazWebResultsGrid_ItemCommand(object source,    System.Web.UI.WebControls.DataGridCommandEventArgs e)
{
  try
  {
    // if user clicks on grid Map button or grid record link
    if (e.Item.ItemType == ListItemType.Item || e.Item.ItemType ==
                  ListItemType.AlternatingItem || e.Item.ItemType ==
                  ListItemType.SelectedItem)
    {
      DataGridItem item = e.Item;

      int findLayerID = (int) Session["LayerID"];

      // item.Cells[1] correspond to column #1 in the grid: LayerID -> column's visibility = false
      // item.Cells[0] correspond to column #0 in the grid: FeatureID -> column's visibility = false
      // USER clicks on Map:  feature selection based on the featureID will be performed
      if (e.CommandName == "FindFeature")
      {
        ZoomToFeature(findLayerID, Convert.ToInt32(item.Cells[0].Text));
      }
        // USER clicks on record link: retrieve feature record based on the featureID will be performed
      else if (e.CommandName == "IDFeature")
      {
        FindFeatureDetails(findLayerID, Convert.ToInt32(item.Cells[0].Text));
      }
    }
  }
  catch (Exception error)
  {
    callErrorPage(String.Format("DataGrid error in {0} command.",
                  e.CommandName), error);
  }
}

private void HazWebResultsGrid_PageIndexChanged(object source,
        System.Web.UI.WebControls.DataGridPageChangedEventArgs e)
{
  try
  {
    // USER clicks on a particular page index
    HazWebResultsGrid.CurrentPageIndex = e.NewPageIndex;

    // Search in all layers, if the field values contians the string
    if (Session["selectedDataSet"] == null)
      return;

    // Retrieve table from session
    HazWebResultsGrid.DataSource = Session["selectedDataSet"] as
                  System.Data.DataTable;
    // Bind it to the DataGrid
    HazWebResultsGrid.DataBind();
  }
  catch (Exception error)
  {
    callErrorPage(String.Format("DataGrid error in {0} page index.",
                  e.NewPageIndex), error);
  }
}
```

```
private void btnFind_Click(object sender, System.EventArgs e)
{
  // Check if distance value is numeric
  if (!Information.IsNumeric(txtDistance.Value))
  {
    Functions.DisplayMessage("Distance value must be numeric");
    return;
  }
  // Check if the distance value is more than 0
  if ((Convert.ToInt32(txtDistance.Value) == 0) ||
                (Convert.ToInt32(txtDistance.Value) < 0))
  {
    Functions.DisplayMessage("Distance value must be more than 0");
    return;
  }

  #region Build Buffers
  // Class Buffers will return a BUFFERS GeometryCollection to be used
  // as the spatial query geometry to find features in the "Find" layer and also returns
  // BUFFERS GraphicElements to be drawn in the MapDescription::CustomGraphics()
  using (WebMap webMap = HazWebMap.CreateWebMap())
  {
    // Remove any polygon elements from the map
    Functions.RemoveAllPolygonElementsFromGraphicsContainer(webMap);

    // Get the server context and map description
    IServerContext serverContext = webMap.ServerContext;
    webMap.ManageLifetime(serverContext);
    IMapDescription mapDescription = webMap.MapDescription;
    webMap.ManageLifetime(mapDescription);
    string mapDescriptionName = mapDescription.Name;

    // Unselect all selected features
    Functions.UnselectAllFeatures(webMap);

    // Get the map server
    IMapServer mapServer = webMap.MapServer;
    webMap.ManageLifetime(mapServer);

    // Create the buffer class
    Buffer createBuffers = new Buffer();
    double distance = Convert.ToDouble(txtDistance.Value);
    // Create buffer
    IGraphicElements graphicBufferElements = createBuffers.CreateBuffer(webMap,
                distance);
    webMap.ManageLifetime(graphicBufferElements);
    if (graphicBufferElements == null)
    {
      Functions.DisplayMessage("Unable to create buffered region");
      return;
    }

    // Change symbology
    IElement polygonElement = createBuffers.ChangeSymbol(webMap);
    webMap.ManageLifetime(polygonElement);

    #region Query features using buffers' GEOMETRY
    int findLayerID = ddlLayers.SelectedIndex + 1;
    ISpatialFilter spatialFilter =
                serverContext.CreateObject("esriGeodatabase.SpatialFilter") as
                ISpatialFilter;
    webMap.ManageLifetime(spatialFilter);
    spatialFilter.Geometry = polygonElement.Geometry;
    // Spatial Filter requires the find layer (feature class) Geometry Field Name
    spatialFilter.GeometryField = Functions.GetGeometryFieldName(webMap,
                findLayerID);

    // Pass selection method to the spatial filter
    spatialFilter.SpatialRel = esriSpatialRelEnum.esriSpatialRelIntersects;
    IQueryFilter findQueryFilter = spatialFilter as IQueryFilter;
    webMap.ManageLifetime(findQueryFilter);
    findQueryFilter.SubFields = "*";

    // Get "Find" Layer -> LayerDescription in order to draw FOUND features
    ILayerDescription layerDescription =
```

```
                mapDescription.LayerDescriptions.get_Element(findLayerID);
webMap.ManageLifetime(layerDescription);

// Query "Find" Layer
IFIDSet fidSet = mapServer.QueryFeatureIDs(mapDescriptionName, findLayerID, findQueryFilter);
webMap.ManageLifetime(fidSet);

// Save index to session
Session["LayerID"] = findLayerID;

if (fidSet.Count() != 0)
{
  // To draw FOUND features use yellow color
  IRgbColor rgb = Converter.ToRGBColor(serverContext,
                        System.Drawing.Color.Green);
  webMap.ManageLifetime(rgb);
  layerDescription.SelectionColor = rgb as IColor;
  layerDescription.SelectionFeatures = fidSet;

  // Store the fidSet into a session
  string sFidSet = serverContext.SaveObject(fidSet);
  Session["FidSet"] = sFidSet;

  //if (Session["OIDField"] != null)
  //{
  // Retrieve FOUND features RECORDSET to populate DataGrid
  IRecordSet recordSet = mapServer.QueryFeatureData(mapDescriptionName,
                        findLayerID, findQueryFilter);
  webMap.ManageLifetime(recordSet);

  // Convert the recordset to a System.Data.DataSet from where
  // the OBJECTID and Display Field are going to be extracted as a table
  // that it is going to bind to the DataGrid
  System.Data.DataSet selectedDataSet =
                        webMap.ConvertRecordSetToDataSet(recordSet, true, false);

  #region Bound/unbound subset table to the DataGrid
  if (selectedDataSet != null)
  {
    HazWebResultsGrid.CurrentPageIndex = 0;

    // Add columns
    if (!this.DynamicResultsColumnAdded)
    {
      this.AddResultsColumns();
    }

    // Change returned table columns name to match datagrid column names
    // Change OBJECTID field name to FeatureID column name
    System.Data.DataTable dataGridTable = selectedDataSet.Tables[0];

    if (dataGridTable.Rows.Count > 0)
    {
      string oidField = Functions.GetOIDFieldName(webMap, findLayerID);
      DataColumn OIDDataColumn = dataGridTable.Columns[oidField];

      string displayField = Functions.GetDisplayField(webMap,
                        findLayerID);
      DataColumn displayFieldDataColumn =
                        dataGridTable.Columns[displayField];

      // Change table's OBJECTID column name to match dataGrid column's name
      if (OIDDataColumn != null)
      {
        OIDDataColumn.ColumnName = DATAGRID_OBJECTID;
      }

      // If the DisplayField is same as OBJECTID
      ButtonColumn btnColumn = HazWebResultsGrid.Columns[1] as ButtonColumn;
      if (OIDDataColumn.ColumnName == displayFieldDataColumn.ColumnName)
      {
        btnColumn.DataTextField = DATAGRID_OBJECTID;
      }
      else
      {
        if (btnColumn.DataTextField == DATAGRID_OBJECTID)
```

```
            {
              btnColumn.DataTextField = DATAGRID_DISPLAYFIELD;
            }

            // Change table's DisplayField column name to match dataGrid column's name
            displayFieldDataColumn.ColumnName = DATAGRID_DISPLAYFIELD;
          }
        } // dataGridTable.Rows.Count > 0

        Session["selectedDataSet"] = dataGridTable;
        HazWebResultsGrid.DataSource = dataGridTable;
        HazWebResultsGrid.DataBind();
        HazWebResultsGrid.Visible = true;
        lblNumberFound.Text = fidSet.Count().ToString() + " " +
                              Functions.GetLayerName(webMap, findLayerID) + " features found";
        lblNumberFound.Visible = true;

        // Check the check box
        if (!chkBuffer.Checked)
        {
          chkBuffer.Checked = true;
        }
        // Hide the table results
        if (tableResults.Visible)
        {
          tableResults.Visible = false;
        }
      }
      else // selectedDataset == null
      {
        HazWebResultsGrid.DataSource = null;
        HazWebResultsGrid.Visible = false;
        lblNumberFound.Text = fidSet.Count().ToString() + " " +
                              Functions.GetLayerName(webMap, findLayerID) + " features found";
        lblNumberFound.Visible = true;
      } // end selectedDataset != null check
      #endregion
    }
    else // fidSet.Count() == 0
    {
      // Hide controls
      HazWebResultsGrid.DataSource = null;
      HazWebResultsGrid.Visible = false;
      tableResults.Visible = false;
      lblNumberFound.Text = fidSet.Count().ToString() + " " +
                            Functions.GetLayerName(webMap, findLayerID) + " features found";
      lblNumberFound.Visible = true;
    } // end fidSet.Count()
    #endregion

    if (graphicBufferElements != null)
    {
      // Before adding new graphic element, remove any old ones first
      Functions.RemoveAllPolygonElementsFromGraphicElements(webMap);

      // Get custom graphics
      IGraphicElements newGraphicElements = mapDescription.CustomGraphics;
      webMap.ManageLifetime(newGraphicElements);
      // Add the new polgon element to the graphic elements
      newGraphicElements.Add((IGraphicElement) polygonElement);

      // Store the XML representation of the BUFFER GraphicElements
      // so the user can turn then on/off without re-creating those
      string graphicElementsXML =
                            serverContext.SaveObject(newGraphicElements);
      Session["Buffers"] = graphicElementsXML;

      // Add the new graphic and refresh map
      mapDescription.CustomGraphics = newGraphicElements;
      webMap.Refresh();
    }

    // Enable the check box control
    if (!chkBuffer.Enabled)
    {
      chkBuffer.Enabled = true;
```

```
        }

        // Close server context
        serverContext.ReleaseContext();
      }
      #endregion
    }

    private void chkBuffer_CheckedChanged(object sender, System.EventArgs e)
    {
      try
      {
        string buffers = Session["Buffers"] as string;
        string sFidSet = Session["FidSet"] as string;

      if ((buffers != null) && (sFidSet != null))
        {
          #region Turn ON/OFF buffers
          using (WebMap webMap = HazWebMap.CreateWebMap())
          {
            IMapDescription mapDescription = webMap.MapDescription;
            IServerContext serverContext = webMap.ServerContext;
            webMap.ManageLifetime(mapDescription);
            if (webMap.IsPooled)
            {
              webMap.ManageLifetime(serverContext);
            }

            if (chkBuffer.Checked)
            {
              // Add graphic element back to the custom graphics
              mapDescription.CustomGraphics = serverContext.LoadObject(buffers)
                                    as IGraphicElements;
              // Apply selected features
              IRgbColor rgb = Converter.ToRGBColor(serverContext, System.Drawing.Color.Green);
              webMap.ManageLifetime(rgb);
              mapDescription.LayerDescriptions.get_Element(
                Convert.ToInt32(ddlLayers.SelectedIndex) + 1).SelectionColor =
                                    rgb as IColor;
              mapDescription.LayerDescriptions.get_Element(
                Convert.ToInt32(ddlLayers.SelectedIndex) + 1).SelectionFeatures = serverContext.LoadObject(sFidSet) as IFIDSet;
            }
            else
            {
              // Remove any polygon elements from the graphics container
              //Functions.RemoveAllPolygonElementsFromGraphicsContainer(webMap);
              Functions.RemoveAllPolygonElementsFromGraphicElements(webMap);
              Functions.UnselectAllFeatures(webMap);
              //mapDescription.CustomGraphics = null;
            }
            // Refresh the map
            webMap.Refresh();
          }
          #endregion
        }
        else
        {
          // Should not come here but if so, remove any polygons
          using (WebMap webMap = HazWebMap.CreateWebMap())
          {
            Functions.RemoveAllPolygonElementsFromGraphicsContainer(webMap);
          }
        }
      }
      catch (Exception error)
      {
        callErrorPage("Error changing buffers visibility.", error);
      }
    }

    #endregion
  }
}
```

APPENDIX E

ERROR CLASS PRINTOUT

```
using System;

namespace HazWeb
{
  public class ErrorPage : System.Web.UI.Page
  {
    // Before deploying application, set showTrace to false
    // to prevent web application users from seeing error details
    private bool showTrace = true;

    protected System.Web.UI.WebControls.Label lblTitle;
    protected System.Web.UI.WebControls.Label lblExtendedMessage;
    protected System.Web.UI.WebControls.Label lblError;

    private void Page_Load(object sender, System.EventArgs e)
    {
      //get error message stored in session
      string message = (string)Session["ErrorMessage"];

      //get details of error from exception stored in session
      string errorDetail = "";
      Exception exception = Session["Error"] as Exception;
      if (exception != null)
      {
        switch (exception.GetType().ToString())
        {
          case "System.UnauthorizedAccessException":
            UnauthorizedAccessException errorAccess = exception as
                                        UnauthorizedAccessException;
            If (errorAccess.StackTrace.ToUpper().
                                        IndexOf("SERVERCONNECTION.CONNECT") > 0)
              errorDetail = "Unable to connect to server. <br>" ;
            break;
        }
        errorDetail += exception.Message;
      }

      //create response and display it
      string response;
      if (message != null && message != "")
        response = String.Format("{0}<br>{1}", message, errorDetail);
      else
        response = errorDetail;
      lblError.Text = response;
      if ((showTrace) && (exception != null))
        lblExtendedMessage.Text = exception.StackTrace;
    }

    #region Web Form Designer generated code
    override protected void OnInit(EventArgs e)
    {
      //
      // CODEGEN: This call is required by the ASP.NET Web Form Designer.
      //
      InitializeComponent();
      base.OnInit(e);
    }

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
      this.Load += new System.EventHandler(this.Page_Load);
    }
    #endregion
  }
}
```

APPENDIX F

FUNCTIONS CLASS PRINTOUT

```
using System;
using System.Data;
using System.Drawing;
using System.Collections;
using System.Web.UI.WebControls;
using ESRI.ArcGIS.Server.WebControls;
using ESRI.ArcGIS.Carto;
using ESRI.ArcGIS.Geometry;
using ESRI.ArcGIS.Server;
using ESRI.ArcGIS.Geodatabase;

namespace HazWeb
{
  /// <summary>
  /// Common functions
  /// </summary>
  public class Functions
  {
    #region Unselect all selected features
    public static void UnselectAllFeatures(WebMap webMap)
    {
      IMapDescription mapDesc = webMap.MapDescription;
      webMap.ManageLifetime(mapDesc);
      for (int i = 0; i < mapDesc.LayerDescriptions.Count; i++)
      {
        ILayerDescription layerDesc = mapDesc.LayerDescriptions.get_Element(i);
        webMap.ManageLifetime(layerDesc);
        layerDesc.SelectionFeatures = null;
      }
    }
    #endregion

    #region Display Message function
    public static void DisplayMessage(string sMessage)
    {
      if (sMessage != "")
      {
        System.Web.HttpContext.Current.Response.Write("<SCRIPT
                    LANGUAGE=\"JavaScript\">\n");
        System.Web.HttpContext.Current.Response.Write("alert(\"" + sMessage +
                    "\")\n");
        System.Web.HttpContext.Current.Response.Write("</SCRIPT>\n");
      }
    }
    #endregion

    #region Zoom To Roads extent
    public static void ZoomToRoadsExtent(WebMap webMap)
    {
      IMapDescription mapDescription = webMap.MapDescription;
      webMap.ManageLifetime(mapDescription);

      // Get the AllRoadsGeo83 layer
      IMap map = webMap.Map;
      for (int i = 0; i < map.LayerCount; i++)
      {
        if ("Fire Threat".ToUpper() == map.get_Layer(i).Name.ToUpper())
        {
          // Get the area of interest of the layer
          ILayer roadsLayer = map.get_Layer(i);
          webMap.ManageLifetime(roadsLayer);
          IEnvelope areaOfInterest = roadsLayer.AreaOfInterest;
          webMap.ManageLifetime(areaOfInterest);

          // Get the full extent of the map
          IMapExtent mapExtent = mapDescription.MapArea as IMapExtent;
          webMap.ManageLifetime(mapExtent);
          mapExtent.Extent = areaOfInterest;
          mapDescription.MapArea = mapExtent as IMapArea;

          // Refresh the map
          webMap.Refresh();
        }
      }
    }
    #endregion
```

```
#region Remove any polygon elements from the graphic elements collection
public static void RemoveAllPolygonElementsFromGraphicElements(WebMap webMap)
{
  // Get the map description
  IMapDescription mapDesc = webMap.MapDescription;
  webMap.ManageLifetime(mapDesc);

  // get the graphic elements
  IGraphicElements gE = mapDesc.CustomGraphics;
  webMap.ManageLifetime(gE);
  ArrayList arrList = new ArrayList();
  // Check if the graphic element is null
  if (gE != null)
  {
    // Check if there are any elements
    if (gE.Count > 0)
    {
      for (int i = 0; i < gE.Count; i++)
      {
        // Get the polygon elements and save the its index
        IElement element = gE.get_Element(i) as IElement;
        webMap.ManageLifetime(element);
        if (element.Geometry.GeometryType ==
                                    esriGeometryType.esriGeometryPolygon)
        {
          arrList.Add(i);
        }
      }

      // Check if there is anything in the arraylist
      if (arrList.Count > 0)
      {
        for (int j = 0; j < arrList.Count; j++)
        {
          // Remove the polygon element
          gE.Remove((int)arrList[j]);
        }
      }
    } // gE.Count > 0
  } // gE != null
  RemoveAllPolygonElementsFromGraphicsContainer(webMap);
}
#endregion

#region Remove any polygon elements from the graphics container
public static void RemoveAllPolygonElementsFromGraphicsContainer(WebMap webMap)
{
  // Get the map
  IMap map = webMap.Map;
  webMap.ManageLifetime(map);

  // Get the graphics container
  IGraphicsContainer gc = map as IGraphicsContainer;
  webMap.ManageLifetime(gc);
  gc.DeleteAllElements();
}
#endregion

#region Find Layer's OID field
public static string GetOIDFieldName(WebMap webMap, int LayerIndex)
{
  // Get the map
  IMap map = webMap.Map;
  webMap.ManageLifetime(map);

  // Get the layer based on the index
  IFeatureLayer featureLayer = map.get_Layer(LayerIndex) as IFeatureLayer;
  webMap.ManageLifetime(featureLayer);
  // Get the feature class associated with the layer
  IFeatureClass featureClass = featureLayer.FeatureClass;
  webMap.ManageLifetime(featureClass);

  // Loop through the feature layer's fields collection
  for (int i = 0 ; i < featureClass.Fields.FieldCount; i++)
  {
```

```
      // Check if field is geometry
      IField field = featureClass.Fields.get_Field(i);
      webMap.ManageLifetime(field);
      if (field.Type == esriFieldType.esriFieldTypeOID)
      {
        return field.Name;
      }
    }
    return string.Empty;
  }
  #endregion

  #region Get Feature Layer's Geometry field name
  public static string GetGeometryFieldName(WebMap webMap, int LayerIndex)
  {
    // Get the map
    IMap map = webMap.Map;
    webMap.ManageLifetime(map);

    // Get the layer based on the index
    IFeatureLayer featureLayer = map.get_Layer(LayerIndex) as IFeatureLayer;
    webMap.ManageLifetime(featureLayer);
    // Get the feature class associated with the layer
    IFeatureClass featureClass = featureLayer.FeatureClass;
    webMap.ManageLifetime(featureClass);

    // Loop through the feature layer's fields collection
    for (int i = 0 ; i < featureClass.Fields.FieldCount; i++)
    {
      // Check if field is geometry
      IField field = featureClass.Fields.get_Field(i);
      webMap.ManageLifetime(field);
      if (field.Type == esriFieldType.esriFieldTypeGeometry)
      {
        return field.Name;
      }
    }
    return string.Empty;
  }
  #endregion

  #region Get Layer's Display field
  public static string GetDisplayField(WebMap webMap, int LayerIndex)
  {
    // Get the map
    IMap map = webMap.Map;
    webMap.ManageLifetime(map);

    // Get the layer based on the index
    IFeatureLayer featureLayer = map.get_Layer(LayerIndex) as IFeatureLayer;
    webMap.ManageLifetime(featureLayer);

    return featureLayer.DisplayField;
  }
  #endregion

  #region Get Layer's name
  public static string GetLayerName(WebMap webMap, int LayerIndex)
  {
    // Get the map
    IMap map = webMap.Map;
    webMap.ManageLifetime(map);

    // Get the layer based on the index
    ILayer layer = map.get_Layer(LayerIndex);
    webMap.ManageLifetime(layer);

    return layer.Name;
  }
  #endregion

  #region Create Table results and create rows to results table
  public static void CreateTableResults(System.Data.DataTable ResultsDataTable, ref System.Web.UI.WebControls.Table
tableResults)
  {
    // Re-creates the table results between postbacks
```

```
  try
  {
    if (ResultsDataTable != null)
    {
      DataRow dataRow = ResultsDataTable.Rows[0];
      string celValue = "";

      foreach (DataColumn dataColumn in ResultsDataTable.Columns)
      {
        TableRow tableRow = new TableRow();

        // Add a two cells to the table
        // Field Name
        TableCell tableCellField = new TableCell();
        tableCellField.Font.Name = "Verdana";
        // If the user wants to use points
        //tableCellField.Font.Size = FontUnit.Point(10);
        tableCellField.Font.Size = FontUnit.XSmall;
        tableCellField.Text = dataColumn.ColumnName;
        // Field Value
        TableCell tableCellValue = new TableCell();
        tableCellValue.Font.Name = "Verdana";
        // If the user wants to use points
        //tableCellField.Font.Size = FontUnit.Point(10);
        tableCellField.Font.Size = FontUnit.XSmall;
        tableCellValue.Text = Convert.ToString(dataRow[dataColumn]);

        // Add Field Name
        tableRow.Cells.Add(tableCellField);

        // Add Field Value
        celValue = tableCellValue.Text;
        if (celValue.ToUpper().IndexOf("HTTP",0) > -1)
        {
          tableCellValue.Text = "<a href=" + celValue + ">" + celValue;
          tableRow.Cells.Add(tableCellValue);
        }
        else
        {
          tableRow.Cells.Add(tableCellValue);
        }

        tableResults.Rows.Add(tableRow);
      }
    }
  }
  catch {}
}
#endregion

#region Is HazWeb Map visible
public static string IsMapVisible(ESRI.ArcGIS.Server.WebControls.Map HazWebMap)
{
  // Creates a javascript map control visibility variable
  // this will make HTML controls visible or invisible according to the map control's visibility
  System.Text.StringBuilder sb = new System.Text.StringBuilder();

  sb.Append("\n" + "<script language=\"javascript\">" + "\n");
  sb.Append("var isMapVisible = " + HazWebMap.Visible.ToString().ToLower() +
                    ";\n");
  sb.Append("</script>" + "\n");

  return sb.ToString();
}
#endregion

#region Search Features and return a Data Table oject
public static DataTable SearchFeature(ESRI.ArcGIS.Server.WebControls.Map HazWebMap, int layerID, int featureID)
{
  try
  {
    using (WebMap webMap = HazWebMap.CreateWebMap())
    {
      string mapName = webMap.MapDescription.Name;

      IMapServer mapServer = webMap.MapServer;
```

```
      #region Get Layer's fields
      IMapServerInfo mapServerInfo = mapServer.GetServerInfo(mapName);
      IMapLayerInfo mapLayerInfo =
                                  mapServerInfo.MapLayerInfos.get_Element(layerID);
      IFields fields = mapLayerInfo.Fields;
      webMap.ManageLifetime(mapServerInfo);
      webMap.ManageLifetime(mapLayerInfo);
      webMap.ManageLifetime(fields);
      #endregion

      #region Find feature
      IMapServerData mapServerData = mapServer as IMapServerData;
      IFeature feature = mapServerData.GetFeature(mapName, layerID,
                                  featureID);
      webMap.ManageLifetime(feature);
      #endregion

      #region Create a DataTable with the Feature attributes
      DataTable dataTable = new DataTable();
      DataRow dataRow = dataTable.NewRow();

      int j = 0;
      for (int i = 0; i < fields.FieldCount; i ++)
      {
        IField field = fields.get_Field(i);
        webMap.ManageLifetime(field);
        if (field.Type != esriFieldType.esriFieldTypeOID && field.Type !=
                                  esriFieldType.esriFieldTypeGeometry)
        {
          dataTable.Columns.Add(new DataColumn(field.AliasName,
                                  typeof(string)));
          dataRow[j] = mapServerData.GetFeatureValue(mapName, layerID,
                                  feature, field.Name);
          j ++;
        }
      }

      dataTable.Rows.Add(dataRow);

      return dataTable;
      #endregion
    }
  }
  catch
  {
    return null;
  }
}
#endregion

#region Remove any point elements from the graphic elements collection
public static void RemoveAllPointElementsFromGraphicElements(WebMap webMap)
{
  // Get the map description
  IMapDescription mapDesc = webMap.MapDescription;
  webMap.ManageLifetime(mapDesc);

  // get the graphic elements
  IGraphicElements gE = mapDesc.CustomGraphics;
  webMap.ManageLifetime(gE);
  ArrayList arrList = new ArrayList();
  // Check if the graphic element is null
  if (gE != null)
  {
    // Check if there are any elements
    if (gE.Count > 0)
    {
      for (int i = 0; i < gE.Count; i++)
      {
        // Get the polygon elements and save the its index
        IElement element = gE.get_Element(i) as IElement;
        webMap.ManageLifetime(element);
        if (element.Geometry.GeometryType ==
                                  esriGeometryType.esriGeometryPoint)
        {
```

```
            arrList.Add(i);
          }
        }

        // Check if there is anything in the arraylist
        if (arrList.Count > 0)
        {
          for (int j = 0; j < arrList.Count; j++)
          {
            // Remove the polygon element
            gE.Remove((int)arrList[j]);
          }
        }
      } // gE.Count > 0
    } // gE != null
  }
  #endregion
 }
}
```

APPENDIX G

HELP PAGE PRINTOUT

DATA ENTRY

Address Section

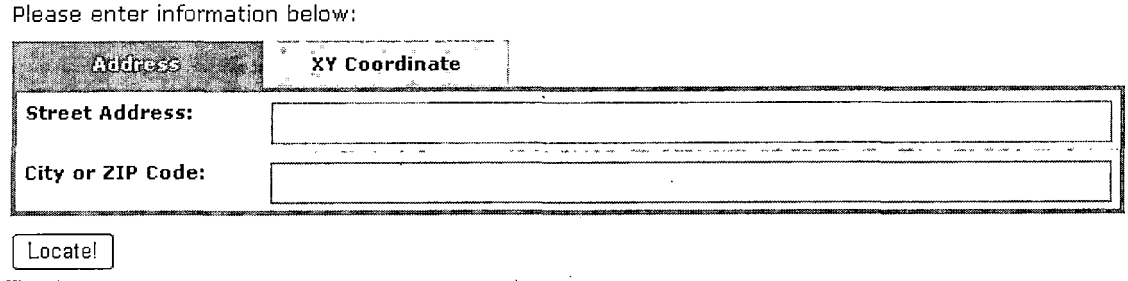


Fig. 1

You have four options when entering data into the *Address* section:

1. An address with a zip code.
2. An address with a city name.
3. A zip code.
4. A city name.

After entering the appropriate address, simple click the *Locate!* button which will take you to the next page displaying the address entered

XY Coordinate Section

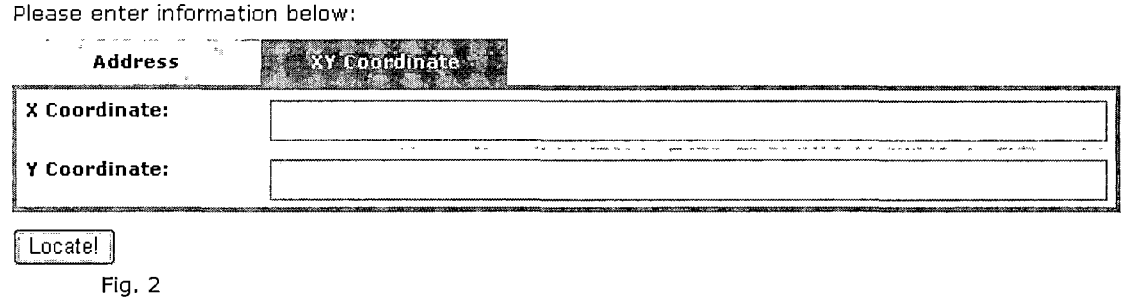


Fig. 2

For the *XY Coordinate* section, you are given the option of entering a valid X and Y coordinate.

1. Enter your X coordinate
2. Enter your Y coordinate
3. Click *Locate!*

DISPLAYING DATA

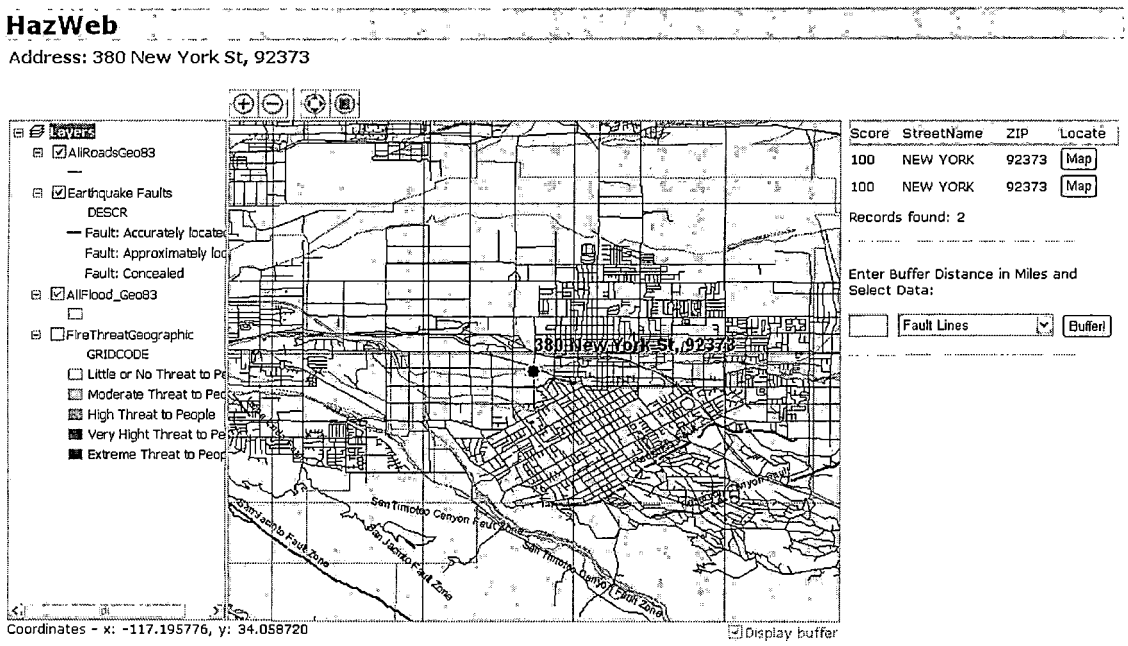


Fig. 3

The address you entered will be displayed with a point and text within the map.

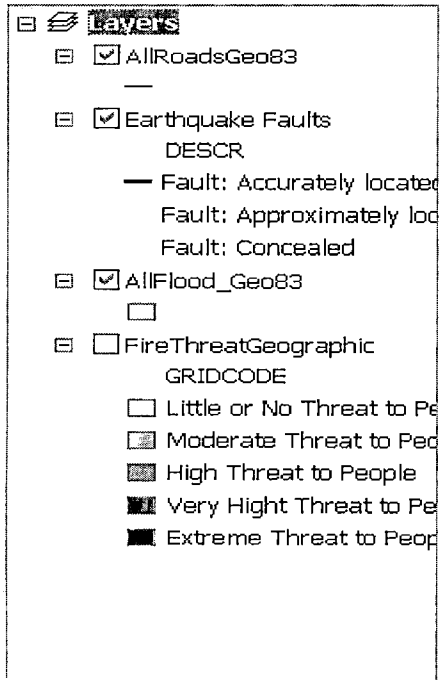


Table of Contents (Fig. 4)

The Table of Contents offers you the ability to display data.

To display data, simply check the check box next to the data you wish to display.

To not display data, simply un-check the check box next to the data you wish to not display.

Navigation Tools

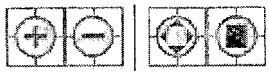


Fig. 5

 The *Zoom In* tool offers you the ability to zoom to a specific location in the map:

1. Activate the tool by clicking on it.
2. Click and drag to create a rectangle on the map.
3. The map will then be zoomed to that location

 The *Zoom Out* tool offers you the ability to zoom out from a specified location in the map:

1. Activate the tool by clicking on it.
2. Click and drag to create a rectangle on the map.
3. The map will then be zoomed out from that location.

 The *Pan* tool offers you the ability to pan around the map:

1. Activate the tool by clicking on it.
2. Click and drag within the map, which will drag the map.

 The *Full Extent* tool zooms the map to the extent of the roads data:

1. Simply click on the *Full Extent* tool.

Creating a Buffered Region

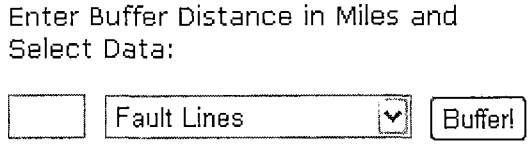


Fig. 6

You can create a buffered region:

1. Enter the mileage you want to create the buffer with in the *Distance* text box.
2. Select the data you want to create the buffer with. Your options are:
    A. Fault Lines
    B. Flood Zones
    C. Fire Threat
3. Click *Buffer!*

Display/Hide Buffered Region

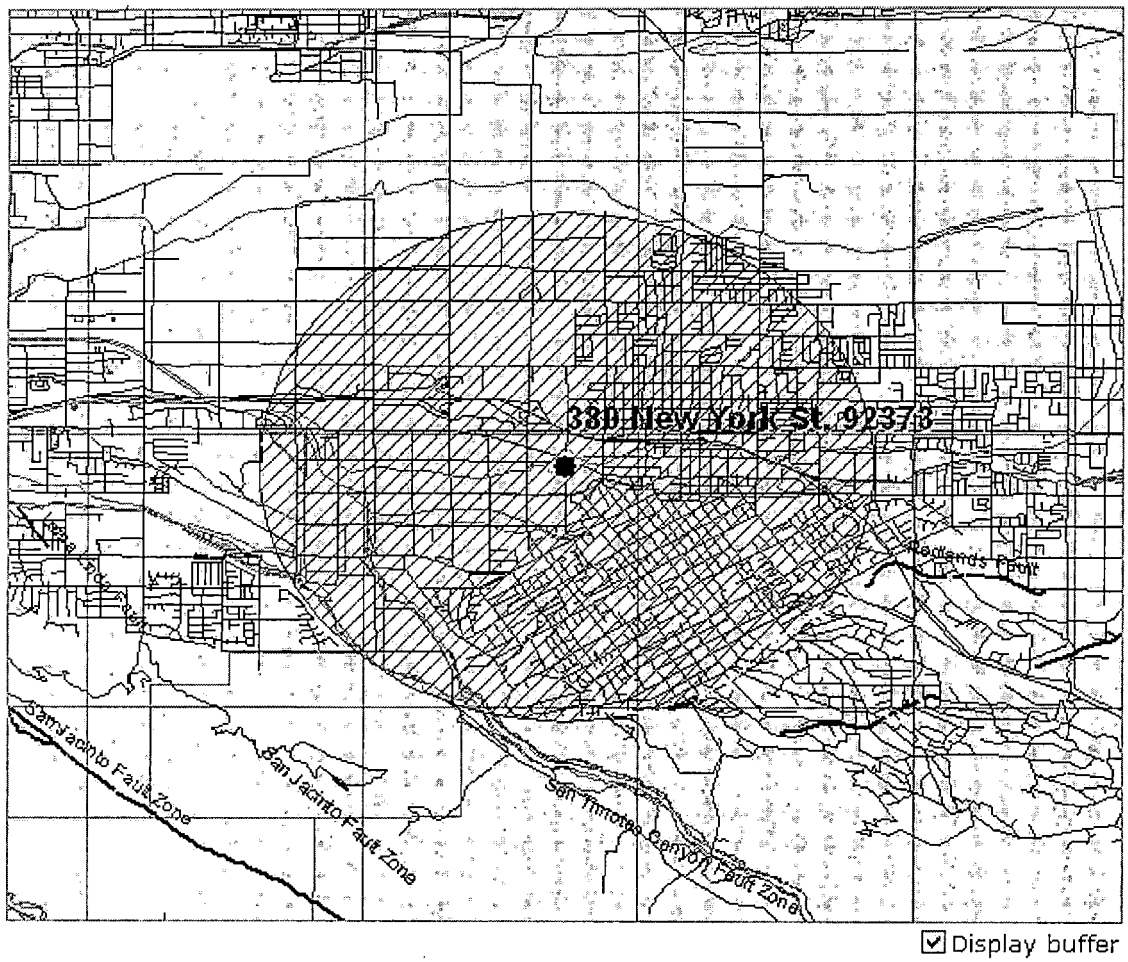


Fig. 7

You can display or hide the buffered region by checking or un-checking the *Display buffer* check box control.

Viewing Found Features

Features found	Locate Feature
Redlands Fault	Map
Redlands Fault	Map
Redlands Fault	Map
Redlands Fault	Map
Redlands Fault	Map

1 2 3 4

18 Earthquake Faults features found

Fig. 8

Field	Value
FNODE_	31338
TNODE_	31586
LPOLY_	0
RPOLY_	0
LENGTH	729.038148
SBFAULT_	4409
SBFAULT_ID	4409
L_SYMB	50
L_NAME	Redlands Fault
DESCR	High-angle fault, scarp, normal slip; accurately located

Fig. 9

After clicking the *Buffer!* Button, a table (Fig. 8) is displayed indicating the number of features found in the buffered region.

Clicking on one of the links in that table (Fig. 8) will display another table directly below it (Fig. 9). This new table will have all the attributes for the selected record.

Clicking the *Map* button in the table (Fig. 8) will zoom to that selected feature for further examination.

REFERENCES

1. Allan Laframboise and Lisa Markham. Developing Applications with ArcGIS Server, .NET (DAAS). ESRI Educational Services, Redlands CA, 2004.

2. Badar, Cameron, et al. ArcGIS Server Administration and Developer Guide. ESRI Press, Redlands CA, 2005.

3. Deitel, Deitel, et al. C#: How to Program. Prentice Hall, New Jersey, 2002.

4. GIS Dictionary. <http://support.esri.com/index.ftm?fa=knowledgebase.gisdictionary.gateway>

5. Jess Liberty and Dan Hurwitz. Programming ASP.NET. O'Reilly, Sebastopol CA, 2003.

6. Keith C. Clarke. Getting Started with Geographic Information Systems. Prentice Hall, New Jersey, 1997.

7. Kelly Hutchins and Jess Borrevik. Advanced ArcObjects using .NET (AAON). ESRI Educational Services, Redlands CA, 2004.

8. Wikipedia: The Free Encyclopedia. <http://en.wikipedia.org/wiki/unit_testing>

9. Wikipedia: The Free Encyclopedia. <http://en.wikipedia.org/wiki/integration_testing>

10. Wikipedia: The Free Encyclopedia. <http://en.wikipedia.org/wiki/system_testing>

11. Street data accessed at ESRI Sample datasets.

12. Flood zone data accessed at GIS Data Depot. <http://data.geocomm.com>

13. Fire threat data accessed at California Department of Forestry and Fire Protection. <http://frap.cdf.ca.gov/data/frapgisdata/select.asp>

14. Earthquake fault line data accessed at United States Geological Survey GeoScience Data Catalog. <http://geo-nsdi.er.usgs.gov>